SIEMENS
NIXDORF

SINIX

# INFORMIX-SQL V4.0

Nachschlagen

Benutzerhandbuch

DOPPEL

# Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
Manualredaktion STM QM 2
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (0 89) 6 36-4 04 43

email im EUnet:
man@sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

# Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
Manualredaktion STM QM 2
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (0 89) 6 36-4 04 43

email im EUnet:
man@sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

# INFORMIX-SQL (SINIX)

Nachschlagen



Ausgabe Dezember 1990 (INFORMIX-SQL V4.0)

# . . . und Schulung?

Zu dem nachstehend beschriebenen Produkt, wie zu fast allen DV-Themen, bieten unsere regionalen Training Center in Berlin, Essen, Frankfurt, Hannover, Hamburg, München, Mainz, Stuttgart, Wien und Zürich Kurse an.

**Auskunft und Info-Material:**

Systemfamilien 7 · 500 und 8890 **Telefon (0 89) 6 36-4 89 87**
Ein- und Mehrplatzsysteme **Telefon (0 89) 6 36-4 24 80**

Siemens Nixdorf Training Center
Postfach 83 09 51, W-8000 München 83



Herausgegeben von
Siemens Nixdorf Informationssysteme AG

# Vorwort

INFORMIX ist ein relationales Datenbanksystem, das auf dem Computer-Markt weit verbreitet ist. INFORMIX bietet

- eine komfortable Menüoberfläche
- leistungsfähige Format- und Listengeneratoren
- eine Datenbanksprache, die den ANSI-Standard erfüllt
- die Möglichkeit zur Gestaltung einer individuellen Menüoberfläche

Das vorliegende Handbuch *INFORMIX-SQL Nachschlagen* ist das Benutzerhandbuch zum Datenbanksystem INFORMIX.

**Welche Vorkenntnisse benötigen Sie?**

Für die Arbeit mit INFORMIX benötigen Sie Grundkenntnisse in der Bedienung des Betriebssystems. Diese vermittelt das entsprechende Betriebssystem-Handbuch.

Für den INFORMIX-Anfänger empfiehlt sich besonders das Handbuch *INFORMIX-SQL Kennenlernen*, das anhand zahlreicher Beispiele und Übungen in das Funktionsangebot von INFORMIX einführt.

Nicht unmittelbar für den Umgang mit INFORMIX nötig, jedoch wichtig für den logischen Entwurf Ihrer Datenbank: Es gibt allgemeine Kriterien für Datenbank-Design, nach denen Sie Ihr Problem Schritt für Schritt in einen sinnvollen Datenbankentwurf umsetzen können.
Zum Thema Datenbank-Design finden Sie Informationen in zahlreichen Büchern. Darüber hinaus bietet die Schule für Kommunikations- und Datentechnik zu diesem Thema Kurse an.

**Was hat sich geändert?**

Das Änderungsprotokoll auf den nachfolgenden Seiten gibt Ihnen einen Überblick über fachliche und strukturelle Änderungen gegenüber dem letzten Ausgabestand.

# Änderungsprotokoll

## Allgemeine Änderungen

### ANSI-Datenbanken

INFORMIX-SE und INFORMIX-ONLINE unterstützen ANSI-Datenbanken.

Eine ANSI-Datenbank hat folgende Merkmale:

- Es ist automatisch Transaktionssicherung eingeschaltet. Die Transaktionssicherung kann nicht ausgeschaltet werden.
- Die voreingestellte Isolationsstufe bei INFORMIX-ONLINE ist Repeatable Read. Das führt dazu, daß sehr viele Sperren gesetzt werden, da auch für Leseoperationen Sperren benötigt werden.
- Fremde Datenbankobjekte müssen mit dem Namen des Eigentümers qualifiziert werden.
- Beim Erstellen einer Tabelle einer ANSI-Datenbank erhält nur der Eigentümer Zugriffsrechte auf die Tabelle. Alle Zugriffsrechte, auch für PUBLIC, müssen explizit mit GRANT vergeben werden.
- Bei Anweisungen, die den ANSI-Standard verletzen, werden Warnungen ausgegeben.

### ANSI-Datenbank erstellen

Sie können eine ANSI-Datenbank erstellen, indem Sie bei CREATE DATABASE die Klausel MODE ANSI angeben.

### Überprüfen auf ANSI-Kompatibilität

Sie können über die Umgebungsvariable DBANSIWARN oder über den Schalter **-ansi** im **isql**-Aufruf und im **saceprep**-Aufruf eine automatische Überprüfung auf ANSI-Kompatibilität einschalten.

**Neue Datentypen**

| | |
|---|---|
| DATETIME | Zeitpunkte |
| INTERVAL | Zeitspannen |

nur für INFORMIX-ONLINE:

| | |
|---|---|
| VARCHAR | Zeichenketten variabler Länge |
| TEXT | Texte beliebiger Länge |
| BYTE | binäre Blob-Daten |

## PERFORM/FORMBUILD

PERFORM ermöglicht Bildschirmgrößen größer als 24 Zeilen/80 Spalten. Die Definition von Farbattributen und Grafiksteuerzeichen ist möglich. Für die Bearbeitung langer CHAR-Felder gibt es einen neuen Multiline-Editor. Die Ausgabe kann jetzt auch im UNLOAD-Format erfolgen.

Kompatibilität: PERFORM kann übersetzte Formatprogramme ab V2.1 verarbeiten. Ein in V4.0 übersetztes Formatprogramm kann jedoch nur in der Version V4.0 benutzt werden. Ältere Versionen können solche Formatprogramme nicht benutzen.

## ACE

Über ACE kann man jetzt auch Sätze aus einer ASCII-Datei verarbeiten.

**Transaktions-Menü**

Das Transaktionsmenü wird ausgegeben, wenn ein Benutzer folgende Aktionen durchführt, ohne zuvor seine Transaktion zu beenden:

- Datenbank eröffnen
- Formatprogramm starten
- Benutzermenu aufrufen
- INFORMIX beenden

Über das Transaktions-Menü wird der Benutzer aufgefordert, seine Transaktion zu beenden.

**Terminologieänderungen**

| | | |
|---|---|---|
| RDSQL | → | SQL |
| Feld (bei Tabellen) | → | Spalte |
| Feldeigenschaft | → | Attribut |

**Umstrukturierung**

Folgende Themen sind jetzt im zentralen SQL-Handbuch [1] beschrieben:

| *Thema* | *INFORMIX Nachschlagen Kapitel in V2.1* | *SQL Kapitel in V4.0* |
|---|---|---|
| Beschreibung der SQL-Anweisungen | 2.3 | 6 |
| Konzepte | 7 | 2 |
| Datentypen und Konventionen | 9 | 3, 4, A.5 |
| Systemtabellen | A.2 | A.2 |

Folgende Kapitel wurden im vorliegenden Handbuch umgestellt:

| *Thema* | *Kapitel in V2.1* | *Kapitel in V4.0* |
|---|---|---|
| INFORMIX-Menüs | 6 | 1.3 |
| Systemumgebung | 8 | 7 |
| Dienstprogramme | A.3 | A.2 |
| ASCII-Tabelle | A.4 | A.3 |

## Änderungen im Einzelnen

### Kapitel 2 SQL anwenden

neu: zusätzliche Zeichen zur Kommentarkennzeichnung: # und - (ANSI)

### Kapitel 3.3 Aufbau eines Format-Quellprogramms

*DATABASE-Abschnitt:*

neu: Angabe einer fremden Datenbank für Anwender der INFORMIX-Netzprodukte

*SCREEN-Abschnitt:*

neu: Maximale Größe der Bildschirmseite innerhalb eines SCREEN-Abschnitts festlegen (SCREEN SIZE zeilen BY spalten).

neu: Grafikzeichen zur Verbesserung des Formatlayouts.

*TABLES-Abschnitt:*

neu: Vergabe eines Aliasnamens für Tabellennamen

neu: Angabe einer externen Tabelle für Anwender der INFORMIX-Netzprodukte.

*ATTRIBUTES-Abschnitt:*

neu: Attribut COLOR: Darstellung des Bildschirmfeldinhalts in Farbe, wahlweise abhängig von einer Bedingung, wahlweise kombinierbar mit anderen Eigenschaften.

neu: Attribut WORDWRAP [COMPRESS]: Darstellung eines mehrzeiligen Bildschirmfeldes.

neu: Attribut PROGRAM: Programm vereinbaren, mit dem Bildschirmfelder der INFORMIX-ONLINE-Datentypen TEXT und BYTE bearbeitet werden.

**Kapitel 4 PERFORM**

neu: Suchoperator ".." zur Bereichsangabe für die Datentypen DATETIME und INTERVAL (Funktion *Suchen*, Abschnitt 4.5.1).

neu: Funktion *Blob*: Anzeige von BLOB-Datentypen (Abschnitt 4.5.3)

neu: Multiline-Editor zum Bearbeiten von WORDWRAP-Feldern (Abschnitt 4.4.4).

neu: Ausgabe in Datei, Ausgabeform wahlweise im Bildschirm-Format oder im UNLOAD-Format (PRINT, Abschnitt 4.5.11).

**Kapitel 5 ACE**

*DATABASE-Abschnitt:*

neu: Schlüsselwort ASCII kann anstelle eines Datenbanknamens angegeben werden, wenn READ-Abschnitt verwendet wird.

*DEFINE-Abschnitt:*

neu: ASCII-Angabe definiert die Name und Datentyp der einzelnen Spalten einer Eingabedatei, die im READ-Abschnitt verwendet wird.

*SELECT-Abschnitt:*

neu: Angabe des Eigentümers im Tabellennamen bei ANSI-Datenbanken

*READ-Abschnitt:*

neuer Abschnitt: Anstelle des SELECT-Abschnitts verwendbar. Liest Sätze aus einer Eingabedatei, die im UNLOAD-Format aufgebaut ist.

*FORMAT-Abschnitt, Ausdrücke für Anweisungen*

neu: Ausdruck CURRENT: aktuelles Datum mit Uhrzeit
neu: Ausdruck WORDWRAP: mehrzeilige Ausgabe mit Umbruch

**Kapitel 6 Benutzermenüs**

Dieses Kapitel ist neu: Über die Funktion *Benutzer-Menue* im Hauptmenü können Sie eigene Benutzermenüs definieren.

**Kapitel 7 Systemumgebung**

*Abschnitt 7.2 Umgebungsvariablen*

| | |
|---|---|
| neu: DBANSIWARN | Überwachung des ANSI-Standards |
| neu: DBMENU | Standard-Benutzermenü vereinbaren |
| neu: INFORMIXTERM | Bildschirmsteuerung termcap/terminfo |
| neu: SQLEXEC | INFORMIX-Backend bestimmen |
| neu: TBCONFIG | tbconfig-Datei für INFORMIX-ONLINE |

*Abschnitt 7.3.1 isql-Aufruf mit Menüs*

| | |
|---|---|
| neu: Angabe **-ansi** | prüft Abweichungen gegenüber ANSI-Standard |
| neu: Angabe **-b** | Aufruf von Benutzermenüs |

*Abschnitt 7.3.3 sformbld*

neu: Angabe **-l** zeilen, **-c** spalten für Größe der Bildschirmseite

*Abschnitt 7.3.5 saceprep*

neu: Angabe **-ansi** prüft Abweichungen gegenüber ANSI-Standard.

*Abschnitt 7.5 Einträge in die TERMINFO*

neu: Anstelle von termcap kann terminfo verwendet werden.

## A.2 Dienstprogramme

neue Dienstprogramme für INFORMIX-SE: dbimport, dbexport, dblog dbexport und dbimport waren bereits in der V2.1 mit dem Backend INFORMIX-TURBO verfügbar.

neue Dienstprogramme für INFORMIX-ONLINE: tbcheck, tblog, tbload, tbunload Diese Dienstprogramme finden Sie im ONLINE-Handbuch [4] beschrieben.

Das Dienstprogramm dbupdate entfällt.

*A.2.2 dbexport*

neu: Angabe **-f** datei schreibt die erzeugten SQL-Anweisungen in die angegebene Datei auf Platte, während die Datendateien auf Band ausgegeben werden.

*A.2.3 dbimport*

neu: Angabe **-f** datei bezeichnet eine Datei, die SQL-Anweisungen für den Datenbankaufbau enthält.

neu: Schalter **-l** für Transaktionsprotokollierung.

neu: Angabe **-ansi** erzeugt eine ANSI-Datenbank.

*A.2.4 dbload*

wurde umfassend überarbeitet.

# Inhaltsverzeichnis

# 1 Einführung



Dieses Kapitel beschreibt, welche Komponenten INFORMIX bereitstellt, welche Benutzerschnittstellen im Dialogbetrieb vorhanden sind, die INFORMIX-Menüstruktur und die im Handbuch verwendeten Darstellungsmittel.

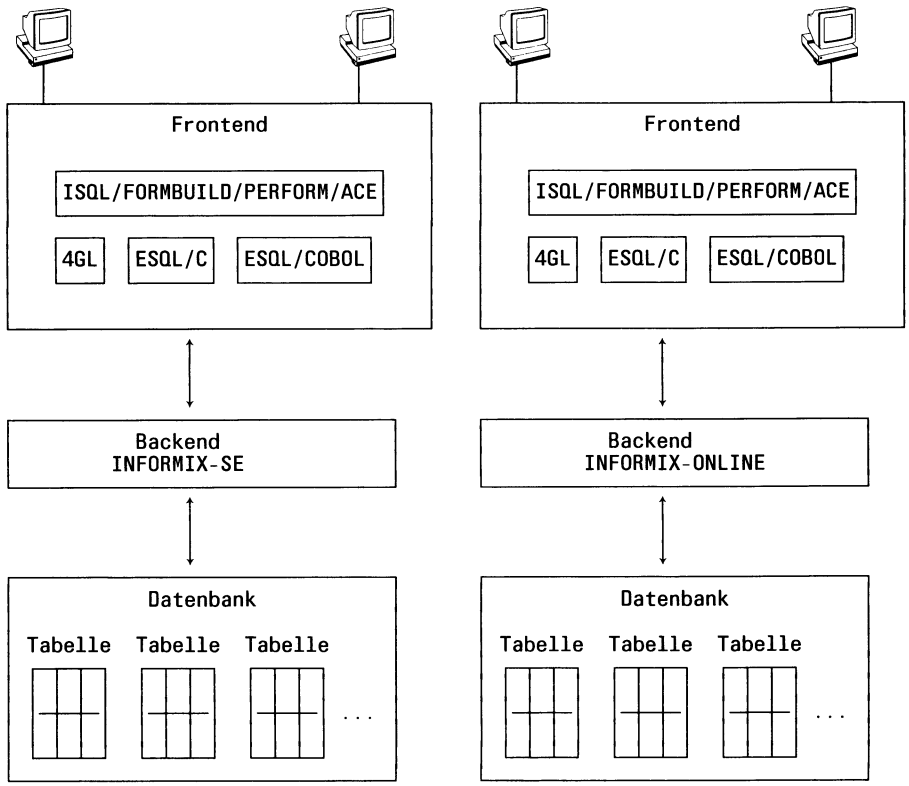
## 1.1 INFORMIX im Überblick

INFORMIX ist ein relationales Datenbanksystem. Ein Datenbanksystem dient dazu, den Benutzer bei der Organisation, Verwaltung und Manipulation großer Datenbestände zu unterstützen. Es übernimmt die Abspeicherung und Bereitstellung der Daten und erledigt Verwaltungsaufgaben. Dazu gehören insbesondere Integritätskontrollen und der Schutz der Daten vor unerlaubten Zugriffen.

Die Besonderheit eines relationalen Datenbanksystems besteht darin, daß der Benutzer die in der Datenbank gespeicherten Daten ausschließlich in Form von Tabellen sieht.

Ein INFORMIX-Datenbanksystem besteht schematisch aus folgenden drei Ebenen:

- Frontend
- Backend
- Datenbank.



### 1.1.1 Frontend

Das Frontend stellt die Benutzerschnittstellen bereit. INFORMIX stellt unterschiedliche Benutzerschnittstellen zur Verfügung. Es gibt Schnittstellen für den Dialogbetrieb und für die Programmeinbettung.

In diesem Handbuch sind die Benutzerschnittstellen für den Dialogbetrieb beschrieben. Abschnitt 1.2 gibt einen Überblick über diese Benutzerschnittstellen.

### 1.1.2 Backend

Das Backend empfängt über ein Frontend die Anweisungen zur Bearbeitung der Datenbank. Es führt die Anweisungen aus und gibt die Ergebnisse an das jeweilige Frontend zurück, wo sie dem Benutzer zur Verfügung gestellt werden.

Das Backend liefert also die Dienste, die zur Ausführung der Datenbankoperationen erforderlich sind. Dazu gehören zum Beispiel die Bereitstellung einer Ablaufumgebung und des benötigten Speicherplatz, sowie Abspeichern und Bereitstellen der Daten.

Bei INFORMIX gibt zwei unterschiedliche Backends:

- INFORMIX-SE (Standard Engine)
- INFORMIX-ONLINE.

**INFORMIX-SE**

INFORMIX-SE organisiert die Datenbank über das SINIX-Dateisystem. Es basiert auf C-ISAM, einer Bibliothek von C-Funktionen. Mit diesen Funktionen werden Dateien erzeugt und verwaltet, in denen die Daten der Datenbank abgespeichert sind. Einzelheiten hierzu finden Sie in Kapitel 2 des SQL-Handbuchs [1].

**INFORMIX-ONLINE**

Das INFORMIX-ONLINE hat eine eigene Plattenverwaltung für seine Datenbanken, unabhängig vom SINIX-Dateisystem. Das hat den Vorteil, daß die Datenhaltung und -verwaltung auf die speziellen Anforderungen eines Datenbanksystems zugeschnitten und daher effizienter und sicherer sind.

Darüberhinaus bietet INFORMIX-ONLINE Möglichkeiten, die bei INFORMIX-SE nicht enthalten sind.
Dazu gehören:

- Bessere Möglichkeiten für die Gewährleistung der Datenintegrität:
  Bei INFORMIX-ONLINE gibt es Isolationsstufen, die festlegen, ob beim Lesen fremde Sperren berücksichtigt und eigene gesetzt werden. Die Isolationsstufe ist einstellbar, so daß jeder Benutzer den Sicherheitsgrad bestimmen kann, der für seine Anweisungen erforderlich ist. Einzelheiten hierzu finden Sie in Kapitel 2 des SQL-Handbuchs [1].

- Bessere Möglichkeiten für die Datensicherung:
  Einen Überblick über die einzelnen Möglichkeiten finden Sie in Kapitel 2 des SQL-Handbuchs [1]. Eine ausführliche Beschreibung der Datensicherungsmethoden enthält das ONLINE-Handbuch [4].

- Datentyp VARCHAR:
  Er ermöglicht es, Spalten zu definieren, die Zeichenketten unterschiedlicher Länge enthalten können. Die ausführliche Beschreibung finden Sie in Kapitel 4 des SQL-Handbuchs [1].

- BLOB (**B**inary **L**arge **O**bjects)-Datentypen BYTE und TEXT:
  Sie ermöglichen es, Spalten zu definieren, die Byte-Folgen beliebiger Länge enthalten können. Die ausführliche Beschreibung finden Sie in Kapitel 4 des SQL-Handbuchs [1].

- Zugriff auf externe Tabellen:
  Sie haben die Möglichkeit, auf Tabellen einer Datenbank zuzugreifen, die nicht die aktuelle Datenbank ist. Einzelheiten hierzu finden Sie in Kapitel 2 des SQL-Handbuchs [1].

- Mehrere INFORMIX-ONLINE-Systeme auf demselben Rechner

**Verträglichkeit von INFORMIX-SE und INFORMIX-ONLINE**

Die beiden Backends können gleichzeitig auf demselben Rechner installiert sein.

Unter Berücksichtigung der genannten Unterschiede können Anwendungen mit jedem der beiden Backends laufen.

Datenbanken, die mit INFORMIX-SE erstellt wurden, müssen konvertiert werden, wenn sie mit INFORMIX-ONLINE verwendet werden sollen, und umgekehrt.

INFORMIX stellt entsprechende Dienstprogramme bereit, über die eine Datenbank von einem Backend in ein anderes Backend importiert werden kann.

**INFORMIX-Netzprodukte**

INFORMIX stellt für die Verwendung entfernter Datenbanken folgende Produkte zur Verfügung:

- INFORMIX-NET für INFORMIX-SE
- INFORMIX-STAR für INFORMIX-ONLINE.

Frontend und Backend können auf verschiedenen Rechnern laufen.

Außerdem haben Sie die Möglichkeit, eine entfernte Datenbank zu erstellen und beim Wechsel der aktuellen Datenbank eine entfernte Datenbank anzugeben.

Bei INFORMIX-ONLINE haben Sie zusätzlich die Möglichkeit, Daten aus externen Tabellen abzufragen, wobei die Datenbank in einem anderen Informixsystem auf demselben oder auf einem anderen Rechner liegen kann.

**Kennzeichnung der Unterschiede**

In diesem Handbuch sind an allen Stellen die Unterschiede zwischen den beiden Backends gekennzeichnet und, falls erforderlich, in einem gesonderten Abschnitt beschrieben.

### 1.1.3 Datenbank

In der Datenbank sind die Benutzer- und Verwaltungsdaten gespeichert. Die einzelnen Datenbankkonzepte für die Verwendung von INFORMIX-Datenbanken sind in Kapitel 2 des SQL-Handbuchs [1] beschrieben.

## 1.2 Benutzerschnittstellen für den Dialogbetrieb

INFORMIX bietet für den Dialogbetrieb folgende Benutzerschnittstellen:

**ISQL:** Zentrale Komponente, über die Sie sämtliche INFORMIX-Funktionen ausführen können. ISQL beinhaltet eine komfortable Menüoberfläche, über die Sie eine Datenbank mit SQL-Anweisungen bearbeiten können, Benutzermenüs erstellen können und auch Format- und Listengeneratoren bedienen können. Einen Überblick über die INFORMIX-Menüstruktur gibt Abschnitt 1.3. Darüberhinaus können Sie ISQL auch ohne Menüführung verwenden (siehe Kapitel 7.3.2).

**FORMBUILD** und **PERFORM:** Komponenten des Formatgenerators. Mit FORMBUILD entwerfen Sie eine Bildschirmmaske für die Dateneingabe und -ausgabe. Der Entwurf wird als Formatprogramm gespeichert. Über PERFORM starten Sie dieses Formatprogramm und bedienen die Bildschirmmaske. FORMBUILD und PERFORM können Sie sowohl über eigene Programmaufrufe (siehe Kapitel 7) als auch über die zentrale Komponente ISQL verwenden.

**ACEPREP** und **ACEGO:** Komponenten des Listengenerators ACE. Mit ACEPREP entwerfen Sie eine Liste für die Datenausgabe. Der Entwurf wird als Listenprogramm abgespeichert. Mit ACEGO starten Sie das Listenprogramm. ACE können Sie sowohl über eigene Programmaufrufe (Kapitel 7) als auch über die zentrale Komponente ISQL verwenden.

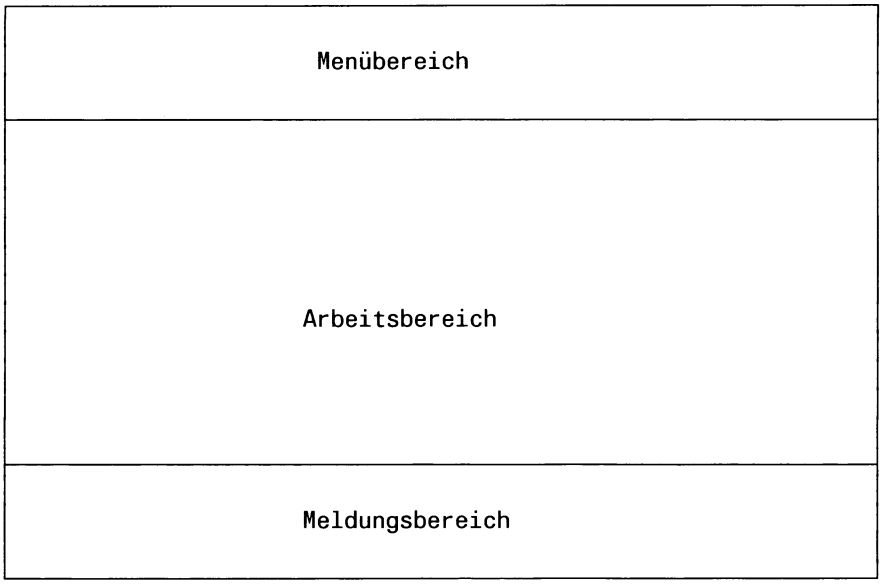
## 1.3 Die INFORMIX-Menüs

Dieses Kapitel

- beschreibt Aufbau und Bedienung der INFORMIX-Menüs
- gibt einen Überblick über die INFORMIX-Menüstruktur

### 1.3.1 Aufbau und Bedienung der INFORMIX-Menüs

Ein INFORMIX-Menü besteht aus 3 Bereichen: dem Menübereich, dem Arbeitsbereich und dem Meldungsbereich:



Der **Menübereich** enthält das Funktionsangebot eines bestimmten Menüs. Es gibt zwei verschiedene Menütypen, die sich in Ihrem Aufbau unterscheiden:

1. Funktionsauswahl-Menü: Menü, in dem Sie eine der angezeigten Funktionen auswählen und ausführen.

```
menüname: 1. funktion 2. funktion 3. funktion 4. funktion 5. funktion ...
kommentar
```

| | |
|---|---|
| menüname | Name des Menüs |
| 1.funktion<br>2.funktion<br>usw. | Funktionen, die Sie in diesem Menü auswählen können. |
| ... | steht am Ende einer Funktionszeile, wenn es eine Folgezeile mit weiteren Funktionen gibt. |
| kommentar | kurzer Erläuterungstext zur momentan markierten Funktion. |

2. Texteingabe-Menü: Menü, in dem Sie durch Eingabe eines Textes im Menübereich entweder einen Namen definieren oder auswählen. Dieser Menütyp erscheint manchmal anschließend an ein Funktionsauswahl-Menü.

```
menüname >> eingabe
kommentar
```

```
auswahl1
auswahl2
...
```

| | |
|---|---|
| menüname | Name des Menüs |
| >> | kennzeichnet den rechts liegenden Bereich als Texteingabebereich. |
| eingabe | Hier können Sie Text eingeben, um einen Namen zu definieren oder auszuwählen. |

| | |
|---|---|
| kommentar | kurzer Erläuterungstext zum Menü. |
| auswahl1<br>auswahl2<br>usw. | Diese Anzeige erscheint im Arbeitsbereich, wenn Sie einen Namen auswählen (nicht neu definieren). auswahl1 usw. sind die möglichen Alternativen, die Sie durch Markieren auswählen können. Sie brauchen dann im Feld *eingabe* keinen Text mehr eingeben. |

Die **Trennzeile zwischen Menübereich und Arbeitsbereich** kann folgende Angaben enthalten:

- Name der aktuell eröffneten Datenbank; rechts eingeblendet
- Nummer der aktuellen Bildschirmseite; links eingeblendet

Der **Arbeitsbereich** hat je nach Menü unterschiedliche Bedeutung.

| | |
|---|---|
| Menü FORMAT | Anzeige von Formaten |
| Menü LISTE | Ausgabe von Listen |
| Menü SQL-DIALOG | Eingabe von Dialoganweisungen und Ausgabe für SELECT-Anweisungen |
| Menü BENUTZER-MENUE | Anzeige des Formats, über das ein Benutzermenü erstellt wird. |
| Menüs DATENBANK und TABELLE | Anzeige von Auswahlangaben |

Im **Meldungsbereich** werden INFORMIX- bzw. Betriebssystemmeldungen ausgegeben, die sich jeweils auf die zuvor durchgeführte Aktion beziehen.

### Bedienung der Menüs

Die hier beschriebenen Bedienungshinweise beziehen sich im wesentlichen auf den Menübereich. Wegen der unterschiedlichen Bedeutung des Arbeitsbereichs in den einzelnen Menüs gelten dort auch unterschiedliche Bedienungshinweise. Diese können Sie jeweils dem Kapitel entnehmen, in dem auch die Funktion des Menüs beschrieben ist.

| | |
|---|---|
| [␣] | Durch Drücken der Leertaste bewegen Sie die Leuchtmarke nach **rechts**. In der Kommentarzeile wird die Erläuterung der gerade markierten Funktion angezeigt. Von rechts außen springt die Markierung nach links außen oder: wenn eine Folgezeile mit Funktionen vorhanden ist, wird diese eingeblendet. Dieselbe Funktion, jedoch nicht in allen Menüs garantiert, bieten auch alle Pfeiltasten (Pfeil rechts und Pfeil unten). |
| [⌫] | Durch Drücken der Korrektur-Taste bewegen Sie die Leuchtmarke nach **links**. Sonst analog zu Leertaste. Dieselbe Funktion, jedoch nicht in allen Menüs garantiert, bieten auch alle Pfeiltasten (Pfeil links und Pfeil oben). |
| [↵] | Durch Drücken der Return-Taste führen Sie die markierte Funktion aus. |
| Buchstabe | Wenn Sie den Anfangsbuchstaben einer angezeigten Funktion eingeben, so wird diese Funktion markiert und gleichzeitig ausgeführt. |
| [DEL] | bricht die eingeleitete Aktion ab und verzweigt in das vorhergende Menü. |
| [END] | beendet Menüs, in denen die Funktion END angeboten wird und verzweigt in das übergeordnete Menü. END im Hauptmenü beendet INFORMIX. |
| [HELP]<br>[CTRL] [W] | liefert eine genaue Erläuterung zur momentan markierten Funktion. |
| [CTRL] [R] | baut den Bildschirm neu auf. Diese Funktion benötigen Sie z. B., wenn die Bildschirmanzeige durch Nachrichtentexte überschrieben wird, die andere Benutzer senden. |
| [CTRL] [\] | bricht die aktuelle Funktion ab und beendet INFORMIX. |
| !kommando | unterbricht INFORMIX. Daran anschließend können Sie ein Betriebssystemkommando eingeben. Ist die Ausführung des Kommandos beendet, erscheint die Aufforderung:<br><br>Weiter mit ↵.<br><br>Danach wird INFORMIX fortgesetzt. |

### 1.3.2 Überblick über die INFORMIX-Menüstruktur

Wenn Sie INFORMIX mit *isql* aufrufen, erscheint zunächst das Hauptmenü:

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
```

```
INFORMIX-SQL:  ...  Tabelle  END
```

Über die einzelnen Funktionen des Hauptmenüs rufen Sie weitere Menüs auf:

Format
ruft das Menü FORMAT auf, über das Sie Formate erstellen und bearbeiten können.

```
FORMAT: Ablauf Modifizieren Generieren Neu Compilieren Loeschen  END
```

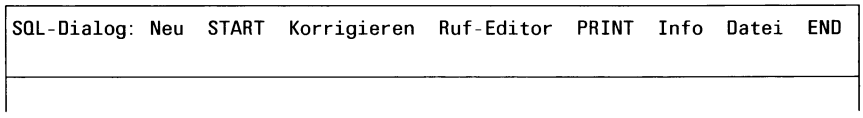
Eine Beschreibung der Funktionen des FORMAT-Menüs finden Sie in Kapitel 3 (FORMBUILD) bzw. Kapitel 4 (PERFORM).

Liste
ruft das Menü LISTE auf, über das Sie Listen erstellen und bearbeiten können.

```
LISTE: Ablauf Modifizieren Generieren Neu Compilieren Loeschen  END
```

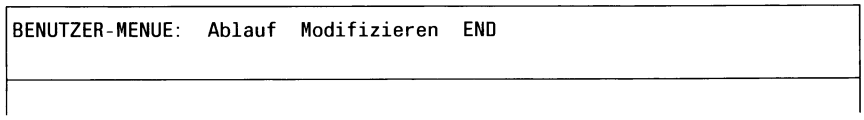
Eine Beschreibung der Funktionen des LISTE-Menüs finden Sie in Kapitel 5 (ACE).

SQL-Dialog
ruft das Menü SQL-DIALOG auf, über das Sie SQL-Anweisungen eingeben können.



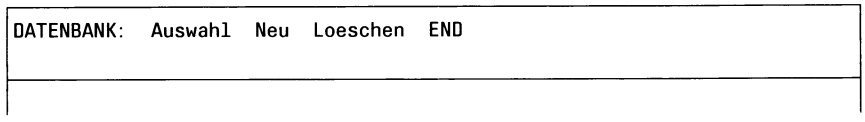

Eine Beschreibung der Funktionen des Menüs SQL-DIALOG finden Sie in Kapitel 2 (SQL anwenden).

Benutzer-Menue
ruft das Menü BENUTZER-MENUE auf, über das Sie Benutzermenüs bearbeiten können.




Eine Beschreibung der Funktionen des Menüs BENUTZER-MENUE finden Sie in Kapitel 6 (Benutzermenüs).

Datenbank
ruft das Menü DATENBANK auf, über das Sie eine Datenbank einrichten, löschen und eröffnen können.



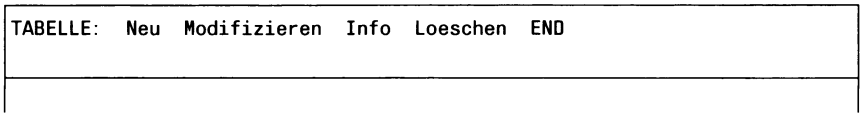

Auswahl
eröffnet eine bereits existierende Datenbank.
Für INFORMIX-SE gilt: Die Datenbank muß im aktuellen Dateiverzeichnis oder in einem Dateiverzeichnis sein, das die Umgebungsvariable DBPATH angibt (siehe Kapitel 7.2). Sie können auch direkt einen Pfadnamen angeben. INFORMIX durchsucht dann nur diesen Pfad.

Neu
erzeugt und eröffnet eine neue Datenbank.

Loeschen
löscht eine Datenbank. Das Löschen ist nur möglich, wenn die Datenbank geschlossen ist.

END
beendet das DATENBANK-Menü und kehrt in das Hauptmenü zurück.

Tabelle
ruft das Menü TABELLE auf, über das Sie eine Tabelle bearbeiten können.




Eine Beschreibung der Funktionen des TABELLE-Menüs, sowie ausführliche Bedienungshinweise finden Sie im Handbuch *INFORMIX Kennenlernen.*

END
beendet INFORMIX-SQL.

## 1.4 Darstellungsmittel

Für die Syntaxdefinitionen in diesem Handbuch verwenden wir folgende Darstelltungsmittel:

| | |
|---|---|
| **Raster** | kennzeichnet Syntaxdefinitionen. |
| GROSS | Schlüsselwort, das sowohl in Klein- als auch in Großbuchstaben angegeben werden kann. |
| **fett** | Schlüsselwort, das in Kleinbuchstaben anzugeben ist. |
| kleinbuchstaben | Parameter, für die Werte angegeben werden müssen. |
| *kursiv* | Parameter, der einen Syntaxteil zusammenfaßt.<br>Diese Darstellung wird verwendet, wenn ein bestimmter Syntaxteil innerhalb der Syntaxdefinition mehrfach vorkommt. |
| [] | In eckige Klammern eingeschlossene Angaben sind optional, d. h. sie dürfen weggelassen werden. Die Klammern dürfen Sie nicht angeben. |
| **[]** | Eckige Klammern in fett müssen angegeben werden. |
| { } | In geschweifte Klammern eingeschlossende Angaben sind alternativ. Die alternativen Angaben sind untereinander aufgelistet. Die Klammern dürfen Sie nicht angeben.<br>*Sonderfall:* Einzeilige geschweifte Klammern fassen Angaben zusammen, auf die sich nachfolgende Wiederholungszeichen (... oder ,...) beziehen. |
| **{ }** | Geschweifte Klammern in fett müssen angegeben werden. |
| Unterstreichung | Die Unterstreichung kennzeichnet einen Standardwert. Der Standardwert wird eingesetzt, wenn keine Alternative angegeben ist. |

| | |
|---|---|
| ... | Wiederholungszeichen. Die unmittelbar vorhergehende Angabe darf beliebig wiederholt werden. Die Wiederholungen sind durch Leerzeichen zu trennen.<br>Bezieht sich das Wiederholungszeichen auf eine Folge von Angaben, so sind diese Angaben durch eine geschweifte Klammer gekennzeichnet.<br>Beispiel: {angabe1␣angabe2␣angabe3}... |
| ,... | wie zuvor; jedoch sind die Wiederholungen durch Komma zu trennen. |
| ␣ | bedeutet, daß mindestens 1 Leerzeichen oder das *Neue-Zeile*-Steuerzeichen zwischen 2 Angaben sein muß. |

# 2 SQL anwenden



SQL ist die Datenbanksprache von INFORMIX, in der die Anweisungen zum Bearbeiten der Datenbank formuliert werden.

Dieses Kapitel beschreibt den Aufruf von SQL und den SQL-Editor.

Die vollständige Beschreibung der Datenbanksprache SQL einschließlich SQL-Anweisungen finden Sie im SQL-Handbuch [1].

### Kommentare in SQL-Anweisungen

SQL-Anweisungen können mit Kommentaren versehen werden. INFORMIX wertet Kommentare nicht aus. Zur Kennzeichnung eines Kommentars benutzen Sie wahlweise folgende Zeichen:

#, --, { }

Schreibweise:

# *kommentar* oder
-- *kommentar* (ANSI-Schreibweise) oder
{ *kommentar* }

## 2.1 Aufruf von SQL

SQL können Sie entweder über das INFORMIX-Menüsystem aufrufen oder direkt auf Betriebssystemebene.

### 2.1.1 Aufruf über INFORMIX-Menüs

Dazu wählen Sie im Hauptmenü die Funktion *SQL-Dialog* aus:

```
INFORMIX-SQL: Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank ...
Fuehren eines interaktiven Dialoges mit SQL-Anweisungen.
```

Falls Sie noch keine Datenbank ausgewählt haben, erscheint erst das Menü:

```
AUSWAHL DATENBANK >>
Datenbank auswaehlen oder Namen eingeben. Weiter mit ↵
```

Sie können jetzt eine Datenbank auswählen oder einfach das Menü abbrechen (Taste DEL). Danach erscheint das Menü SQL-DIALOG.

**Menü SQL-DIALOG**

```
SQL-DIALOG: Neu  START  Korrigieren  Ruf-Editor  PRINT  Info  Datei  END
```

Neu
: ruft den SQL-Editor auf (siehe Abschnitt 2.2) und löscht den zuletzt im Editor gespeicherten Inhalt. Jetzt können Sie Ihre SQL-Anweisungen eingeben. Wenn Sie mehrere Anweisungen angeben wollen, so müssen Sie diese durch ein Semikolon (;) trennen. Die Anweisungen bleiben im Editor gespeichert, bis sie mit *Neu* wieder gelöscht werden oder bis INFORMIX beendet wird.

START
: führt die aktuellen SQL-Anweisungen aus.

Korrigieren
ruft den SQL-Editor auf (siehe Abschnitt 2.2) und zeigt den zuletzt gespeicherten Inhalt. Damit können Sie bequem Syntaxfehler korrigieren oder auch die aktuellen Anweisungen abändern.

Ruf-Editor
ruft den für INFORMIX definierten Editor des Betriebssystems auf. Der zuletzt im SQL-Editor gespeicherte Inhalt wird dort automatisch angezeigt. Wenn Sie keinen Dateinamen vereinbaren, erzeugt INFORMIX einen temporären Dateinamen. Mit der Umgebungsvariablen DBEDIT (siehe Kapitel 7.2) können Sie den Ruf-Editor für INFORMIX festlegen.

PRINT
führt die aktuellen SQL-Anweisungen aus und leitet die Ausgabe anschließend weiter:

- an den Drucker,
- in eine Datei oder
- über Pipe an ein Programm.

Info
gibt Informationen über Tabellen aus. Folgende Informationen sind möglich:

- Name und Datentyp der Spalten
- Indizes
- Zugriffsrechte
- Status-Informationen

Datei
verwaltet Anweisungsdateien. Sie können über diese Funktion:

- SQL-Anweisungen aus einer Anweisungsdatei einlesen,
- SQL-Anweisungen in einer Anweisungsdatei abspeichern oder
- eine Anweisungsdatei löschen.

END
beendet das Menü SQL-Dialog und kehrt zurück in das Hauptmenü. Der zuletzt im SQL-Editor gespeicherte Inhalt wird gelöscht.

### 2.1.2 Aufruf auf Betriebssystemebene

Es gibt zwei unterschiedliche Betriebssystem-Aufrufe für SQL:

- Sie umgehen das Hauptmenü, indem Sie gleich beim Aufruf von INFORMIX das Menü SQL-DIALOG auswählen oder
- Sie rufen INFORMIX ohne Menüs auf, um anschließend SQL-Anweisungen einzugeben.

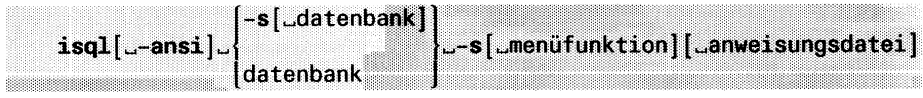
#### Aufruf von INFORMIX mit dem Menü SQL-Dialog

Bei diesem Aufruf rufen Sie INFORMIX auf und verzweigen dabei sofort in das Menü SQL-DIALOG. Außerdem können Sie bereits beim Aufruf

- eine Funktion dieses Menüs auswählen, die dann sofort ausgeführt wird
- eine Anweisungsdatei angeben, die die gewünschten Anweisungen enthält

Sobald Sie das Menü SQL-DIALOG verlassen (END), wird auch das INFORMIX-Programm beendet.

Syntax des Aufrufs:




Eine Beschreibung der Operanden finden Sie in Kapitel 7.3.1.

### Aufruf von INFORMIX ohne Menüs

Bei diesem Aufruf rufen Sie INFORMIX ohne Menüs auf. Dabei können Sie eine Anweisungdatei angeben, die anschließend ausgeführt wird. Sonst geben Sie ihre Anweisungen direkt nach dem Aufruf ein. Mit der Taste END beenden Sie INFORMIX.

Syntax des Aufrufs:


Eine Beschreibung der Operanden finden Sie in Kapitel 7.3.2.

## 2.2 SQL-Editor

INFORMIX stellt Ihnen zur Bearbeitung der SQL-Anweisungen einen eigenen Editor zur Verfügung, den SQL-Editor.

**Merkmale des SQL-Editors**

- Anzeigefenster: 19 Zeilen, je Zeile 79 Zeichen
- Anzeigefenster kann vertikal verschoben werden; horizontales Verschieben ist nicht möglich.
- Sie können nicht über Spalte 79 hinaus schreiben; Zeilen können jedoch länger als 79 Zeichen werden, wenn Sie die Funktion *Einfuegen* verwenden (Taste F15). Eine solche Zeile können Sie dann mit der Funktion *Zeile trennen* (Taste F16) wieder in mehrere Zeilen zerlegen, die innerhalb des Anzeigebereichs liegen.

Anstelle des SQL-Editor können Sie auch über die Funktion *Ruf-Editor* im Menü SQL-DIALOG einen Editor des Betriebssystems verwenden.

**Aufruf des SQL-Editors**

Mit der Funktion *Neu* oder *Korrigieren* rufen Sie den SQL-Editor auf. Danach erhalten Sie folgendes Menü:

```
menu:  START = Ausfuehren     END = Beenden   F15 = Beginn/Ende Einfuegen
       F16   = Zeile trennen  F17 = Loeschen bis Zeilenende
--n-----------------------------------------------------------------------
```

*menu* steht für NEU bzw. KORR.

START — führt die aktuellen SQL-Anweisungen aus und wechselt anschließend in das übergeordnete Menü. Die Anweisungen bleiben weiterhin gespeichert. Bei Fehler wird im übergeordneten Menü die Funktion *Korrigieren* markiert.

END — speichert die aktuellen Anweisungen und wechselt in das übergeordnete Menü. Dort werden die aktuell gespeicherten Anweisungen im Arbeitsbereich angezeigt.

| | |
|---|---|
| F15 | abwechselndes Ein- und Ausschalten des Einfügemodus. Das Einfügen gilt ab Position der Schreibmarke. |
| F16 | trennt die Zeile ab Schreibmarkenposition. Der abgetrennte Teil wird als neue Zeile eingefügt. |
| F17 | löscht die Zeile ab Schreibmarkenposition. |
| *n* | Zeilennummer der Zeile, in der sich die Schreibmarke gerade befindet. |

Weitere Tasten, die Sie verwenden können:

| | |
|---|---|
| ESC | wirkt wie die Taste END |
| CTRL | wirkt wie die Taste F15 |
| CTRL D | wirkt wie die Taste F17 |
| → | bewegt die Schreibmarke nach rechts. |
| ← | bewegt die Schreibmarke nach links. |
| ↑ | bewegt die Schreibmarke nach oben. |
| ↓ | bewegt die Schreibmarke nach unten. |
| ↵ | Schreibmarke springt in die nächste Zeile, Spalte 1. |
| ⇥ | Tabulatorfunktion: Schreibmarke springt nach rechts auf die nächste der Spalten 8, 16, 24, 32 usw. (Schrittweite: 8) |
| ⇤ | Tabulatorfunktion: Schreibmarke springt nach links auf die nächste der Spalten 8, 16, 24, 32 usw. (Schrittweite: 8) |
| ⇧ | Schreibmarke springt um 18 Zeilen nach oben. |
| ⇩ | Schreibmarke springt um 18 Zeilen nach unten. |
| ↖ | Schreibmarke springt nach Zeile 1, Spalte 1. |
| ⌫ | löscht das Zeichen vor der Schreibmarkenposition. |
| CHAR DELETE<br>CTRL X | löscht das Zeichen an der Schreibmarkenposition. |

| | |
|---|---|
| LINE DELETE | löscht die Zeile, in der die Schreibmarke steht. |
| CHAR INSERT | fügt ein Leerzeichen an der Schreibmarkenposition ein. |
| LINE INSERT | fügt eine Leerzeile vor der Zeile ein, in der die Schreibmarke steht. |

# 3 FORMBUILD - Formatprogramm erstellen und übersetzen



Mit FORMBUILD erstellen und übersetzen Sie ein Format(-Quell)programm. Das übersetzte Formatprogramm bildet die Eingabe für PERFORM, das daraus ein Format erstellt (PERFORM, siehe Kapitel 4).

*Was enthält ein Formatprogramm?*

Im Formatprogramm legen Sie Aufbau und Inhalt eines Formats fest. Grundlage für ein solches Formatprogramm sind die Tabellen einer Datenbank. Diese Datenbank und die entsprechenden Tabellen müssen schon bestehen, bevor Sie mit FORMBUILD arbeiten.

Der Aufruf von FORMBUILD ist sowohl über INFORMIX-Menüs, als auch auf Betriebssystemebene möglich.

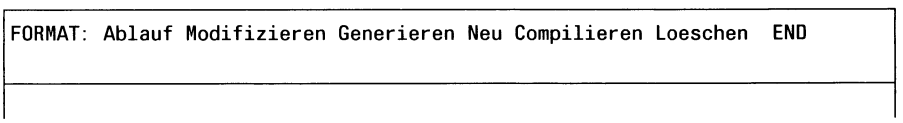
## 3.1 Ein Formatprogramm über INFORMIX-Menüs erstellen

Über die INFORMIX-Menüs können Sie ein Formatprogramm erzeugen, indem Sie entweder

- ein Standard-Formatprogramm erzeugen (siehe Abschnitt 3.1.1) oder
- ein bereits vorhandenes Formatprogramm, z. B. ein Standard-Formatprogramm abändern (siehe Abschnitt 3.1.2) oder
- ein Formatprogramm selbst schreiben (siehe Abschnitt 3.1.4)

In allen Fällen verwenden Sie dazu das Menü FORMAT. Dieses Menü erhalten Sie durch Auswahl der Funktion *Format* im INFORMIX-Hauptmenü.

**Aufbau des Menüs FORMAT**




Ablauf
: PERFORM-Aufruf: startet ein mit FORMBUILD übersetztes Formatprogramm. Am Bildschirm erscheint eine Liste aller zugreifbaren Formatprogramme. Die Datei, die das ablauffähige Formatprogrmm (*.frm*) enthält, muß im aktuellen Dateiverzeichnis oder in dem über die Umgebungsvariable DBPATH vereinbarten Dateiverzeichnis enthalten sein (DBPATH siehe Kapitel 7.2).

Modifizieren
: ändert ein bereits vorhandenes Formatprogramm. Am Bildschirm erscheint eine Liste aller zugreifbaren Formatprogramme. Die Datei, die das Format(-Quell)progrmm (*.per*) enthält, muß im aktuellen Dateiverzeichnis oder in dem über die Umgebungsvariable DBPATH vereinbarten Dateiverzeichnis enthalten sein (DBPATH siehe Kapitel 7.2). Nachdem Sie das gewünschte Formatprogramm ausgewählt haben, ruft FORMBUILD den über die Umgebungsvariable DBEDIT vereinbarten Editor auf und eröffnet dort eine Datei, die das Formatprogramm enthält. Weitere Informationen siehe Abschnitt 3.1.2.

Generieren

erzeugt ein Standard-Formatprogramm für eine oder mehrere Tabellen der aktuell eröffneten Datenbank. Wenn Sie noch keine Datenbank eröffnet haben, erscheint ein Auswahlmenü, das die verfügbaren Datenbanken anzeigt. FORMBUILD fragt ab, welchen Namen das Formatprogramm erhalten soll. Namenskonventionen siehe Abschnitt 3.1.5.

Anschließend gibt FORMBUILD eine Liste aller Tabellen der aktuellen Datenbank aus. Sie wählen die Tabellen aus, für die ein Standard-Format erzeugt werden soll. FORMBUILD erzeugt dazu automatisch ein Standard-Formatprogramm und übersetzt dieses Formatprogramm anschließend. Sie erhalten ein für PERFORM ablauffähiges Formatprogramm. Weitere Informationen siehe Abschnitt 3.1.1.

Neu

Die Funktion *Neu* benötigen Sie, wenn Sie ein Formatprogramm selbst schreiben möchten. FORMBUILD fragt ab, welchen Namen das Formatprogramm erhalten soll (Namenskonventionen siehe Abschnitt 3.1.5). Anschließend wird der über die Umgebungsvariable DBEDIT vereinbarte Editor aufgerufen und eine Datei eröffnet, in Sie das Formatprogramm eingeben können. Weitere Informationen siehe Abschnitt 3.1.4.

Compilieren

übersetzt ein Formatprogramm und erzeugt dadurch ein für PERFORM ablauffähiges Formatprogramm. Am Bildschirm erscheint eine Liste aller zugreifbaren Formatprogramme. Die Datei, die das Format(-Quell)progrmm (*.per*) enthält, muß im aktuellen Dateiverzeichnis oder in dem über die Umgebungsvariable DBPATH vereinbarten Dateiverzeichnis enthalten sein (DBPATH siehe Kapitel 7.2). Nachdem Sie das gewünschte Formatprogramm ausgewählt haben, wird die Übersetzung gestartet. Enthält das Formatprogramm Fehler, so wird das Übersetzen abgebrochen und über ein entsprechendes Menü die Funktion *Korrigieren* angeboten (siehe Abschnitt 3.1.3).

Loeschen
: löscht ein Formatprogramm. Am Bildschirm erscheint eine Liste aller zugreifbaren Formatprogramme. Die Betriebssystemdateien (*.per* und *.frm*) müssen im aktuellen Dateiverzeichnis oder in dem über die Umgebungsvariable DBPATH vereinbarten Dateiverzeichnis enthalten sein, damit das Löschen möglich ist (DBPATH siehe Kapitel 7.2). Nachdem Sie das gewünschte Formatprogramm ausgewählt haben, verlangt FORMBUILD erneut eine Bestätigung, ob das Löschen durchgeführt werden soll.

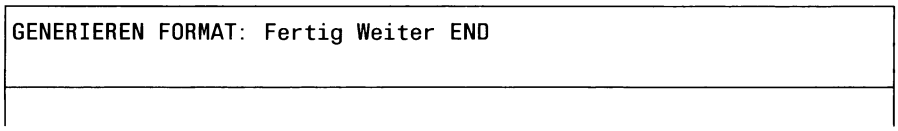
END
: beendet das Menü FORMAT und kehrt zum INFORMIX-Hauptmenü zurück.

### 3.1.1 Ein Standard-Formatprogramm erstellen

In einem Standardformat werden alle Spalten der angegebenen Tabellen untereinander aufgelistet. Jede Tabellenspalte ist genau einem Bildschirmfeld zugeordnet. Bei großen CHAR- oder VARCHAR-Feldern werden einer Tabellenspalte mehrere Bildschirmfelder zugeordnet. Das zugehörige Standard-Formatprogramm wird von FORMBUILD automatisch erstellt und übersetzt, wenn Sie die Funktion *Generieren* des Menüs FORMAT auswählen. Das Standard-Formatprogramm enthält nur ein Minimum an Anweisungen, die unbedingt nötig sind, damit es für PERFORM ablauffähig ist.

*Vorgehensweise*

- Wählen Sie die Funktion *Generieren.*
- Wählen Sie eine Datenbank aus, falls noch keine eröffnet ist.
- Definieren Sie den Namen für das Formatprogramm (Namenskonventionen siehe Abschnitt 3.1.5).
- Wählen Sie die Tabellen aus, für die Sie das Standard-Format erstellen wollen. Maximal 14 Tabellen dürfen in einem Format gleichzeitig verwendet werden. Über die Menüfunktion *Modifizieren* können Sie die Anzahl auf 16 Tabellen erhöhen. Nach jeder Tabellenwahl erscheint das folgende Menü:




Fertig
wählen Sie, sobald die Wahl der Tabellen beendet ist, also keine weitere Tabelle im Format enhalten sein sollen.

Weiter
wählen Sie, wenn Sie weitere Tabellen für das Format auswählen wollen.

END
Rückkehr in das Menü FORMAT.

*Hinweis*

Ein Standard-Formatprogramm bietet eine gute Grundlage für spätere Erweiterungen. Es enthält sozusagen den fehlerfreien formalen Grundaufbau, den jedes Formatprogramm benötigt, um ablaufähig zu sein. Aus diesem Grundprogramm können Sie dann durch ensprechende Erweiterungen Ihr individuelles Format erzeugen.

### 3.1.2 Ein Formatprogramm ändern

Mit der Funktion *Modifizieren* im Menü FORMAT ändern Sie ein bestehendes Formatprogramm:

*Vorgehensweise*

- Wählen Sie die Funktion *Modifizieren* aus.
- Wählen Sie am Bildschirm das Formatprogramm aus, das Sie abändern wollen. FORMBUILD ruft anschließend den über die Umgebungsvariable DBEDIT vereinbarten Editor auf, mit dem Sie das Formatprogramm bearbeiten können. FORMBUILD benutzt für Ihre Änderungen eine Arbeitsdatei. Die Datei *name.per*, die das ursprüngliche Formatprogramm enthält, bleibt unverändert.
- Tragen Sie Ihre Änderungen ein. Halten Sie dabei die Syntax ein, die im Abschnitt 3.3 beschrieben ist.

- Beenden Sie den Editor. Dabei müssen Sie Ihre Änderungen entsprechend der Regeln des Editors sichern. Das geänderte Formatprogramm bleibt zunächst in der Arbeitsdatei gespeichert. Die Betriebssystemdatei *name.per* enthält weiterhin das ursprüngliche Formatprogramm (ohne Änderung).
- Nun erhalten Sie das Menü MODIFIZIEREN FORMAT:

```
MODIFIZIEREN FORMAT: Compilieren Sichern-und-END Verwerfen-und-END
```

Compilieren
: übersetzt das Formatprogramm und erzeugt damit ein für PERFORM ablauffähiges Formatprogramm.

  Ist die Übersetzung erfolgreich abgelaufen, wird die Funktion *Sichern-und-END* angeboten. Erst wenn Sie diese Funktion ausführen, werden die Änderungen in den Dateien *name.per* (Formatprogramm) und *name.frm* (übersetztes Formatprogramm) wirksam. Ohne Sichern enthalten diese Dateien (falls vorhanden) den Zustand vor der Änderung.

  Sind beim Übersetzen Fehler aufgetreten, erhalten Sie das Menü COMPILIEREN FORMAT (siehe Abschnitt 3.1.3).

Sichern-und-END
: sichert das Formatprogramm. Die entsprechenden Betriebssystemdateien *name.per* (Formatprogramm) und *name.frm* (übersetztes Formatprogramm) werden aktualisiert bzw. neu erzeugt, falls noch nicht vorhanden.

Verwerfen-und-END
: verwirft die Änderungen. Die Arbeitsdatei, die das geänderte Formatprogramm enthält, wird gelöscht. Die entsprechenden entsprechenden Betriebssystemdateien *name.per* (Formatprogramm) und *name.frm* (übersetztes Formatprogramm) werden nicht überschrieben.

### 3.1.3 Fehler nach Aufruf der Funktion Compilieren

Falls beim Übersetzen Fehler aufgetreten sind, erscheint folgendes Menü:

```
COMPILIEREN FORMAT: Korrigieren END
```

Korrigieren
ruft den über die Umgebungsvariable DBEDIT vereinbarten Editor auf und zeigt das fehlerhafte Formatprogramm an. Die Fehler sind ensprechend kommentiert. Diese Kommentare müssen Sie nicht löschen. Nach Korrektur beenden Sie den Editor und sichern die Arbeitsdatei nach den Regeln Ihres Editors. Sie erhalten wieder das Menü MODIFIZIEREN FORMAT. Dort wählen Sie erneut die Funktion *Compilieren*, um das korrigierte Formatprogramm übersetzen zu lassen.

END
beendet dieses Menü und kehrt in das vorherige Menü zurück.

### 3.1.4 Ein Formatprogramm selbst schreiben

Um ein für PERFORM ablauffähiges Formatprogramm zu erzeugen, können Sie neben den in den Abschnitt 3.1.1 und 3.1.2 beschriebenen Verfahren auch die Funktion *Neu* im FORMAT-Menü wählen.

Die Funktion *Neu* ruft den in der Umgebungsvariablen DBEDIT vereinbarten Editor auf. Sie müssen dann das gesamte Formatprogramm in der im Abschnitt 3.3 angegebenen Syntax selbst schreiben. Ist das Formatprogramm fertig, so müssen Sie mit der Funktion *Compilieren* des FORMAT-Menüs daraus das für PERFORM ablauffähige Formatprogramm erzeugen.

*Hinweis*

Ein Formatprogramm von Anfang an selbst zu schreiben ist relativ aufwendig. Es empfiehlt sich deswegen, zunächst über die Funktion *Generieren* ein Standard-Formatprogramm zu erzeugen (Abschnitt 3.1.1) und dieses dann mit *Modifizieren* entsprechend abzuändern (Abschnitt 3.1.2).

### 3.1.5 Konventionen

**Name des Formatprogramms**

*Maximale Länge:* 10 Zeichen

*Erlaubte Zeichen:* Buchstaben, Ziffern, _ (Unterstrich). Das erste Zeichen muß ein Buchstabe sein.

*Verbotene Namen:* reservierte Wörter (siehe SQL-Handbuch [1])

**Editor**

Zur Bearbeitung Ihres Formatprogramms stellt Ihnen FORMBUILD den Editor zur Verfügung, der über die Umgebungsvariable DBEDIT definiert ist (DBEDIT siehe Kapitel 7.2).

**Dateien**

FORMBUILD erzeugt für jedes Formatprogramm zwei Betriebssystemdateien:

| | |
|---|---|
| name.*per* | Datei, die das Format(-Quell)programm enthält. |
| name.*frm* | Datei, die das übersetzte Formatprogramm enthält. |

name ist der von Ihnen definierte Name des Formatprogramms.

Diese Dateien müssen entweder im aktuellen Dateiverzeichnis oder in dem Dateiverzeichnis, das die Umgebungsvariable DBPATH festlegt, enthalten sein, damit FORMBUILD und PERFORM darauf zugreifen können.

**Grenzwerte**

Maximale Anzahl Tabellen im Format:

14, wenn Funktion *Generieren* verwendet wird; sonst 16.

Standardgröße einer Bildschirm-Seite (SCREEN-Abschnitt):

Breite: 80 Zeichen
Höhe: 20 Zeilen

Die Größe kann verändert werden (Angabe SIZE im SCREEN-Abschnitt).

## 3.2 Ein Formatprogramm auf Betriebssystemebene erstellen

Es gibt zwei Möglichkeiten, ein Format auf der Betriebssystemebene zu erzeugen:

- entweder über den FORMBUILD-Aufruf *sformbld* oder
- über einen *isql*-Aufruf mit anschließender Auswahl des FORMBUILD-Menüs

Die zweite Möglichkeit, das Erzeugen eines Formats über einen *isql*-Aufruf, finden Sie in Kapitel 7.3.1 beschrieben.

### 3.2.1 sformbld - ein ablauffähiges Formatprogramm erstellen

Auf Betriebssystemebene können Sie mit dem Aufruf *sformbld* ein ablauffähiges Formatprogramm erstellen, indem Sie

- entweder ein Programm für ein Standard-Format erzeugen (Aufruf *sformbld* mit Schalter **d**) oder
- selbst ein Format-Quellprogramm schreiben.

  Wenn Sie ein Formatprogramm selbst schreiben, gehen Sie folgendermaßen vor:

  Sie schreiben mit einem Editor ein Quellprogramm für das Format. Dabei ist es gleichgültig, ob Sie das Programm völlig neu schreiben oder ob Sie ein bereits vorhandenes Quellprogramm abändern. Der Dateiname für das Quellprogramm muß das Suffix *.per* tragen. Das Quellprogramm übersetzen Sie mittels des Programmaufrufs *sformbld.*

  Sofern beim Übersetzen keine Fehler aufgetreten sind, erzeugt *sformbld* ein für PERFORM ablauffähiges Formatprogramm. Dieses ablauffähige Programm trägt das Suffix *.frm.*

  Falls jedoch beim Übersetzen Fehler auftreten, wird eine Fehlerdatei mit dem Suffix *.err* angelegt. Sie müssen die Fehler dann im Quellprogramm verbessern und das verbesserte Programm anschließend erneut compilieren. Bei erfolgreicher Compilierung wird die Fehlerdatei gelöscht.

Die Syntax des Aufrufs lautet:

```
sformbld[␣-q][␣-l␣zeilen][␣-c␣spalten][␣-v]{␣datei}
                                            {␣-d   }
```

**-q** unterdrückt alle Meldungen am Bildschirm.

**-l** zeilen
Maximale Seitenlänge innerhalb des SCREEN-Abschnitts in Zeilen. Davon bleiben 4 Zeilen für das System reserviert. Fehlt die Angabe, so gilt als Standardwert: 24 Zeilen.

**-c** spalten
Maximale Breite des Formats in Spalten. Fehlt die Angabe, so gilt als Standardwert: Spaltenanzahl der längsten Zeile im SCREEN-Abschnitt.

**-v** vergleicht bei Spalten vom Datentyp CHAR die Länge der im SCREEN-Abschnitt definierten Bildschirmfelder mit der Länge der jeweils zugeordneten Tabellenspalten.

Falls Längenunterschiede auftreten, werden Warnungen in eine Datei *datei.err* ausgegeben. Diese Datei *datei.err* dient lediglich zu Ihrer Information über aufgetretene Längenunterschiede. Das Format ist benutzbar, obwohl Längenunterschiede aufgetreten sind. Sie können allerdings durch diese Längenunterschiede Schwierigkeiten bei der Ein- und Ausgabe von Daten bekommen (siehe dazu Abschnitt 3.3, SCREEN-Abschnitt, *feldbegrenzer*).

Falls Sie Längenunterschiede beseitigen wollen, müssen Sie das in der Datei *datei.per* tun. Die Datei *datei.per* müssen Sie nach einer Korrektur erneut übersetzen. Bei erfolgreicher Übersetzung wird die Datei *datei.err* gelöscht.

**-d** erfüllt zwei Aufgaben: Zum einen wird ein Quellprogramm für ein Standard-Format erzeugt. Gleichzeitig wird dieses Quellprogramm übersetzt und so ein ablauffähiges Formatprogramm erzeugt. Die hierfür notwendigen Informationen fragt FORMBUILD im Dialog vom Benutzer ab. FORMBUILD fragt nach dem Namen des zu erstellenden Standard-Formatprogramms, nach dem Namen der Datenbank und nach den Tabellen, die Sie verwenden wollen. Sie dürfen bei diesem Schalter maximal 14 Tabellen angeben. Sie können das Formatprogramm aber auf bis zu 16 Tabellen erweitern.

Als Seitenlänge im SCREEN-Abschnitt generiert FORMBUILD automatisch SCREEN SIZE 20 (s. SCREEN-Abschnitt im Abschnitt 3.3).

datei

muß der Name der Datei sein, die das Format-Quellprogramm enthält. Der Dateiname hat die Endung *.per*, die Sie hier aber nicht anzugeben brauchen.

FORMBUILD übersetzt das Format-Quellprogramm und schreibt das übersetzte Formatprogramm in eine Datei mit dem Suffix *.frm*. Auf diese Datei greift PERFORM zu, wenn ein Format aufgerufen wird.

Falls beim Übersetzen Fehler aufgetreten sind, legt *sformbld* eine Fehlerdatei mit dem Suffix *.err* an. Diese Datei enthält eine Kopie des Quellprogramms mit Fehlerkommentaren. Sie müssen dann die Fehler im Quellprogramm korrigieren und das Programm erneut compilieren. Bei erfolgreicher Übersetzung wird die Fehlerdatei gelöscht.

## 3.3 Aufbau eines Format-Quellprogramms

Im Abschnitt 3.3 wird die Syntax eines Format(-Quell)programms erläutert. Dabei werden Beispiele gegeben, die sich zum größten Teil auf das Beispielprogramm *muster* beziehen. Sie finden das Beispielprogramm im Anhang A.1.

Jedes Formatprogramm muß mindestens die folgenden Informationen enthalten:

- welche Datenbank mit dem Format bearbeitet werden soll,
- wie der Bildschirm des Formats in PERFORM aussehen soll (wo sind die Bildschirmfelder, Feldnamen und Überschriften auf dem Bildschirm angeordnet?),
- welche Tabellen bearbeitet werden sollen und
- welches Bildschirmfeld welcher Tabellenspalte entsprechen soll

Ein Formatprogramm besteht aus fünf Abschnitten. Die ersten vier Abschnitte müssen angegeben werden, nur der letzte, der INSTRUCTIONS-Abschnitt, ist wahlfrei. Auch das Schlüsselwort END ist wahlfrei. Sie können END jedoch zur besseren Strukturierung Ihres Formatprogramms an das Ende jedes Abschnitts setzen.

**Überblick über den Aufbau des Formatprogramms**

| Abschnitt | Inhalt |
|---|---|
| DATABASE | ist der erste Abschnitt des Formatprogramms und enthält den Namen der Datenbank, für die Sie das Format erstellen. |
| SCREEN | folgt dem DATABASE-Abschnitt und enthält das Format-Layout. Sie ordnen hier Bildschirmfelder, Feldnamen und Überschriften so an, wie sie später im FORMAT erscheinen sollen. |
| TABLES | folgt dem SCREEN-Abschnitt und enthält die Namen der im Formatprogramm verwendeten Tabellen. |

| | |
|---|---|
| ATTRIBUTES | folgt dem TABLES-Abschnitt. Hier ordnen Sie Bildschirmfeldern über einen Feldbezeichner Tabellenspalten oder DISPLAYONLY-Felder zu. Darüberhinaus können Sie Attribute für die einzelnen Bildschirmfelder definieren. Sie können ein Bildschirmfeld z.B. mit einem Kommentar versehen oder Standardwerte für ein Feld festlegen. |
| [INSTRUCTIONS] | ist der letzte Abschnitt des Formatprogramms und enthält z.B. Anweisungen für Rechenoperationen oder Definitionen über Beziehungen, die die verwendeten Tabellen zueinander haben sollen. |

### Kommentare im Formatprogramm

Ein Formatprogramm darf Kommentare enthalten. INFORMIX wertet diese Kommentare nicht aus. Kommentare müssen Sie in geschweifte Klammern { } einschließen. Kommentare dürfen außerhalb der geschweiften Klammern des SCREEN-Abschnitts überall dort stehen, wo Leerzeichen erlaubt sind.

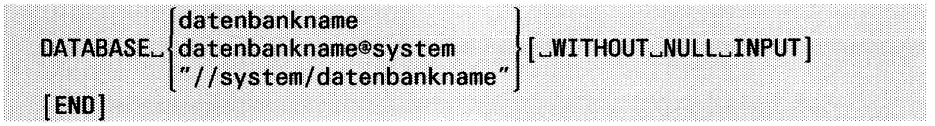
### Verwendung der INFORMIX-ONLINE-Datentypen TEXT und BYTE

*Datentyp TEXT:* Standardmäßig wird für TEXT nur *ein* Bildschirmfeld generiert. Bei der Anzeige des Inhalts in diesem Feld werden führende Leerzeichen berücksichtigt. Mit dem Attribut WORDWRAP können Sie sich Inhalte mehrzeilig ausgeben lassen, das Bearbeiten mit dem Multiline-Editor ist jedoch nicht möglich. Zum Bearbeiten wählen Sie das Attribut PROGRAM, in dem Sie einen Editor vereinbaren können. Ohne PROGRAM steht Ihnen der in DBEDIT definierte Editor zur Verfügung (Umgebungsvariable DBEDIT siehe Kapitel 7.2). Ist DBEDIT nicht gesetzt, so gilt der Editor des Betriebssystems.

*Datentyp BYTE:* Eine Darstellung des Inhalts im Format ist nicht möglich. Felder dieses Typs werden mit der Ausgabe <BYTE Wert> versehen. Die Bearbeitung ist nur mit einem eigens dafür vorgesehenen Programm möglich, das Sie über das Attribut PROGRAM vereinbaren.

## DATABASE-Abschnitt

Im DATABASE-Abschnitt geben Sie die Datenbank an, für die Sie das Format erstellen.




datenbankname
: Name der Datenbank. Wenn Sie INFORMIX-Netzprodukte verwenden (siehe Kapitel 1.1.2), dann können Sie hier auch eine fremde Datenbank angeben. Die fremde Datenbank wird mit dem Namen des INFORMIX-Systems qualifiziert (Angabe *system*). Wenn Sie hier eine ANSI-Datenbank angeben, so müssen evtl. angegebene fremde Tabellen (TABLES-Abschnitt) ebenfalls aus einer ANSI-Datenbank stammen.

system
: bezeichnet das INFORMIX-System. INFORMIX-SE: Name des Rechners, auf dem sich die fremde Datenbank befindet. INFORMIX-ONLINE: Servername, den der INFORMIX-ONLINE-Verwalter bei der Initialisierung vergeben hat.

WITHOUT NULL INPUT
: weist allen Bildschirmfeldern der Datenbank einen Standardwert zu, deren zugeordnete Tabellenspalten mit NOT NULL definiert sind, d.h. keine NULL-Werte erlauben. Welcher Standardwert zugewiesen wird, hängt vom Datentyp der Tabellenspalte ab:

  CHAR: Leerzeichen '␣'
  INTERVAL, numerische Datentypen: Null '0'
  DATE: 31.12.1899
  DATETIME: 1899-12-31␣23:59:59.99999

*Achtung:* WITHOUT NULL INPUT müssen Sie angegeben, wenn Sie in ein Bildschirmfeld des Datentyps CHAR, das mit NOT NULL definiert ist (keine NULL-Werte erlaubt), Leerzeichen '␣'eingeben wollen. WITHOUT NULL INPUT erzeugt automatisch Leerzeichen in solchen Feldern. Eine explizite Eingabe von Leerzeichen in ein solches Feld ist über das Format nicht möglich.

Bildschirmfelder, für die mit dem Attribut DEFAULT (ATTRIBUTES-Abschnitt) ein Standardwert definiert wird, erhalten den Standardwert aus DEFAULT.

## SCREEN-Abschnitt

Im SCREEN-Abschnitt legen Sie das Aussehen des Formates fest, wie es am Bildschirm erscheint. Sie können dabei auch ein Format auf mehrere Bildschirmseiten verteilen, indem Sie entsprechend mehrere SCREEN-Abschnitte definieren.
Zur besseren Gestaltung des Formatlayouts stehen Ihnen Grafiksteuerzeichen zur Verfügung (siehe Abschnitt *Grafikzeichen im Format*).

Folgende Angaben müssen gemacht werden:

Es muß mindestens eine Bildschirmseite definiert werden. Eine Bildschirmseite wird eingeleitet durch das Schlüsselwort SCREEN und eine öffnende Klammer { und beendet durch eine schließende Klammer }.

Über *feldbegrenzer* muß die Länge und Position der Bildschirmfelder definiert werden.

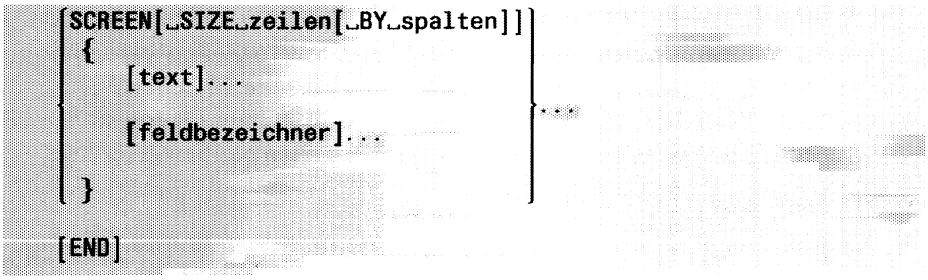
Jedem Bildschirmfeld muß ein *feldbezeichner* zugeordnet werden. Über diesen *feldbezeichner* wird das zugehörige Bildschirmfeld innerhalb des Formatprogramms identifiziert. Jedem *feldbezeichner* muß im ATTRIBUTES-Abschnitt ein oder mehrere Tabellenspalten oder ein DISPLAYONLY-Feld zugeordnet werden.

Darüberhinaus empfiehlt sich folgendes:

Geben Sie jedem Bildschirmfeld einen Feldnamen. Dieser Feldname erscheint im Format beim jeweiligen Bildschirmfeld. Dem Benutzer wird so das Arbeiten mit dem Format erleichtert.

Verteilen Sie das Format gegebenenfalls auf mehrere Bildschirme.

Entwerfen Sie die einzelnen Bildschirme durch Überschriften und graphische Gestaltung benutzerfreundlich.




**Hinweis**

Die fett gedruckten Klammern { } [] müssen angegeben werden.

SCREEN

Im SCREEN-Abschnitt wird die Gestaltung einer Bildschirmseite definiert. Die Definition der Bildschirmseite muß mit dem Schlüsselwort SCREEN beginnen. Wenn Sie für das Format mehrere Bildschirmseiten definieren wollen, so müssen Sie je Bildschirmseite 1 SCREEN-Abschnitt definieren. Die Bildschirmseite innerhalb eines SCREEN-Abschnittes läßt sich widerum in weitere Bildschirmseiten unterteilen (siehe Angabe SIZE).

Beim Erstellen eines Standardformats wird für jede Tabelle ein SCREEN-Abschnitt angelegt.

Über die PERFORM-Menüfunktion *Format* blättern Sie jeweils zur nächsten Bildschirmseite.

SIZE

bezeichnet die maximale Bildschirmgröße innerhalb eines SCREEN-Abschnitts. Diese Angabe muß immer im 1. SCREEN-Abschnitt gemacht werden und gilt dann für alle nachfolgenden SCREEN-Abschnitte.

Die Angabe SIZE ist auch im Betriebssystemaufruf *sformbld* möglich:

```
sformbld -l zeilen -c spalten datei
```

Damit übersetzen Sie das Formatprogramm mit den entsprechenden Angaben neu (Angaben -l zeilen und -c spalten sind wahlfrei), ohne das Formatprogramm (Suffix *.per*) zu ändern. Die ausführliche Beschreibung von *sformbld* finden Sie in Abschnitt 3.2.1.

zeilen

Seitenlänge in Zeilen. Sie können Werte zwischen 6 und 600 angeben. 4 Zeilen bleiben immer reserviert. d.h. wenn Sie 16 Zeilen vereinbaren, beträgt die tatsächliche Länge einer Bildschirmseite 12 Zeilen.
Fehlt die Angabe, so gilt als Standardwert: 24 Zeilen. Davon stehen 20 Zeilen für den Bildschirm zur Verfügung, 4 Zeilen bleiben reserviert.

BY spalten

Seitenbreite in Spalten. Sie können Werte zwischen 30 und 600 angeben. Fehlt die Angabe, so gilt als Standardwert: 80 Spalten.
Ist der im SCREEN-Abschnitt definierte Bildschirm breiter als hier angegeben, so wird die Angabe *spalten* ignoriert. *formbld* gibt lediglich beim Übersetzen eine Warnung aus.

{ }

Jede Bildschirmseite muß in geschweifte Klammern eingeschlossen werden. Jede geschweifte Klammer muß am Anfang und allein in einer neuen Zeile stehen. Eine Seite beginnt mit einer öffnenden Klammer { und wird mit einer schließenden Klammer } abgeschlossen.

text

ist wahlfrei. Hier tragen Sie z.B. Überschriften, Feldnamen und graphische Gestaltungen ein. In einem Standard-Format werden als Feldnamen automatisch die Namen der Tabellenspalten eingetragen. Wenn Sie in *text* Umlaute verwenden, beachten Sie bitte die Hinweise im Anhang A.3 am Ende der ASCII-Tabelle.

[ ]

sind Feldbegrenzer. Die eckigen Klammern *müssen* angegeben werden. Durch sie wird die Länge des Bildschirmfeldes festgelegt.

Die Länge des Bildschirmfeldes ist die Anzahl der Zeichen, die zwischen die Klammern geschrieben werden können.

Der Inhalt einer Tabellenspalte läßt sich auch auf mehrere Bildschirmfelder verteilen. Näheres siehe Angaben *spalte[ganzzahl,ganzzahl]* und WORDWRAP im ATTRIBUTES-Abschnitt.

Mit Ausnahme von WORDWRAP-Feldern (siehe ATTRIBUTES-Abschnitt) sollten die Länge eines Bildschirmfeldes und die Spaltenlänge, wie Sie sie in einer Datenbank-Tabelle definiert haben, gleich sein.

Falls Sie die Länge des Bildschirmfeldes gegenüber der in der Datenbanktabelle definierten Spaltenlänge verkürzen und dann mit einer

INSERT-Anweisung (siehe SQL-Handbuch [1]) Werte eingeben, die länger als die im SCREEN-Abschnitt definierte Feldlänge sind, hat dies folgende Konsequenzen:

- Bei Bildschirmfeldern eines numerischen Datentyps füllt PERFORM dieses Feld mit Sternchen '*'.
- Bei Bildschirmfeldern des Datentyps CHAR schneidet PERFORM das rechte Ende der CHAR-Zeichenkette ab.

Bei Feldern des Datentyps CHAR können Sie die Länge der Bildschirmfelder mit der Länge der jeweils zugeordneten Tabellenspalte vergleichen lassen. Dies geschieht über den Schalter **-v** bei Aufruf *sformbld* (siehe Abschnitt 3.2.1)

Statt eckiger Klammern können Sie für das Anzeigen des Formats auf dem Bildschirm auch andere Zeichen als Feldbegrenzer definieren. Wenn Sie andere Feldbegrenzer verwenden wollen, müssen Sie im INSTRUCTIONS-Abschnitt die Anweisung DELIMITERS geben.

*Bildschirmfelder im Standardformat*

Bei einem Standardformat wird die Länge der Bildschirmfelder von der Spaltenlänge bestimmt, die in der Datenbanktabelle definiert wurde.

Dabei gilt für die Datentypen CHAR und VARCHAR (INFORMIX-ONLINE):
Spalten, die länger als 51 Zeichen sind, werden automatisch im mehrere Bildschirmfelder aufgeteilt.

Beim INFORMIX-ONLINE-Datentyp TEXT wird immer nur *ein* Bildschirmfeld (Länge 58 Zeichen) generiert.

feldbezeichner

geben den Bildschirmfeldern einen Namen. Über diesen Namen identifizieren Sie das Bildschirmfeld im ATTRIBUTES- und INSTRUCTIONS-Abschnitt.

Der Feldbezeichner darf nicht länger als das durch die Feldbegrenzer definierte Bildschirmfeld sein, maximal jedoch 50 Zeichen.

Das erste Zeichen muß ein Buchstabe sein. Der Rest des Feldbezeichners darf Buchstaben, Zahlen und Unterstriche enthalten. Ein Feldbezeichner kann auch aus nur einem Buchstaben bestehen.

FORMBUILD unterscheidet nicht zwischen Groß- und Kleinschreibung. Die Feldbezeichner *a1* oder *A1* sind also für FORMBUILD identisch.

Ein Bildschirmfeld muß einen eindeutigen, von allen anderen Bildschirmfeldern unterscheidbaren Feldbezeichner haben.

Ausnahmen: Sie benutzen einen Feldbezeichner für mehrere Bildschirmfelder,

- wenn Sie eine Tabellenspalte oder ein DISPLAYONLY-Feld in Ihrem Format bewußt an mehr als einer Stelle verwenden wollen. Auf diese Weise ordnen Sie einer bestimmten Tabellenspalte bzw. einem bestimmten DISPLAYONLY-Feld mehrere Bildschirmfelder zu.
- wenn Sie das WORDWRAP-Attribut verwenden (näheres s. ATTRIBUTES-Abschnitt).

*Standardformat:* Dort sind die Feldbezeichner bereits eingefügt. Für Bildschirmfelder, die nur ein Zeichen lang sind, werden als Feldbezeichner automatisch die Buchstaben a bis z vergeben. Daher kann ein Format maximal 26 Bildschirmfelder enthalten, die die nur ein Zeichen lang sind.

**Beispiel zum SCREEN-Abschnitt:**

Sie sehen den SCREEN-Abschnitt des Formatprogramms *muster* (siehe Anhang A.1):

```
screen
{
================================================================================
================================================================================
KUNDENDATEN:
   Kundennummer: [k1          ]
          Firma: [k4                    ]
       Vorname: [k2         ]              Nachname: [k3              ]
       Adresse: [k5                    ]
                [k6                    ]
           Plz: [k7  ]   Ort: [k8              ]     Bundesland: [k9]
       Telefon: [k10           ]
================================================================================
================================================================================
}
screen
{
================================================================================
KUNDENNUMMER: [k1          ]           FIRMA: [k4                    ]
AUFTRAGSDATEN:
 Auftragsnummer: [au11         ]    Auftragsdatum: [au12          ]
  Artikelnummer: [p13         ]       Hersteller: [p16]
   Beschreibung: [ar14             ]               [m17            ]
    Stueckliste: [ar16             ]
                                            Menge: [p18         ]
                                            Preis: [ar15              ]
                                     Gesamtpreis: [p19               ]
LIEFERHINWEIS:
  Fremde Nummer: [au20       ]      Liefergebuehr: [d1          ]
          offen: [a]          Auftragsgesamtsumme: [d2          ]
    Lieferdatum: [au21       ]
      Zahldatum: [au22       ]
  Lieferhinweis: [au23                                      ]
}
end
```

Das Formatprogramm *muster* ist speziell auf die Kundenverwaltung und Auftragsabwicklung ausgerichtet. Im SCREEN-Abschnitt sind zwei Bildschirmseiten definiert. Jede Bildschirmseite wird mit SCREEN und einer öffnenden Klammer { eingeleitet und mit einer schließenden Klammer } beendet.

Die erste Bildschirmseite enthält ausschließlich Spalten der Tabelle *kunde* und dient der Bearbeitung von Kundendaten.

Der zweite Bildschirm enthält Spalten der Tabellen *kunde, auftrag, hersteller, artikel* und *posten* (siehe Anhang A.1). Dieser zweite Bildschirm wird für die Auftragsbearbeitung verwendet.

**Grafikzeichen im Format**

Mit speziellen Steuerzeichen im Formatprogramm können Sie für den Formatbildschirm Linien definieren, z. B. in Form eines Rechtecks, und damit das Layout verbessern. Sie können das gesamte Format in einen Rahmen setzen oder mit Linien unterteilen.

Diese Steuerzeichen schreiben Sie im SCREEN-Abschnitt an die gewünschte Stelle des Bildschirmaufbaus. Folgende Steuerzeichen zur Gestaltung eines Rahmens sind möglich:

| Steuerzeichen | Auswirkung |
|---|---|
| p | erzeugt eine linke obere Ecke |
| q | erzeugt eine rechte obere Ecke |
| b | erzeugt eine rechte untere Ecke |
| d | erzeugt eine linke untere Ecke |
| – | erzeugt eine horizontale Linie |
| \| | erzeugt eine vertikale Linie |

Damit diese Steuerzeichen als Grafiksteuerzeichen erkannt werden, müssen Sie jeweils den Beginn und das Ende einer Steuerzeichenfolge kennzeichnen: Die Zeichenfolge '\g' am Beginn der Steuerzeichenfolge schaltet den Grafikmodus ein, am Ende der Steuerzeichenfolge aus. Die Zeichenfolge '\g' fügen Sie am besten erst dann ein, wenn Ihre SCREEN-Definition fertig ist. Die durch das Einfügen entstandene Verschiebung des Bildschirmaufbaus im Quellprogramm wirkt sich nicht auf das Format aus (siehe Beispiel).

*TERMCAP- und TERMINFO-Dateien*

Beim Ablauf des Formatprogramms holt sich INFORMIX die entsprechenden Grafikzeichen aus der TERMCAP- oder der TERMINFO-Datei (siehe Kapitel 7.4 und 7.5). Dort müssen folgende Variablen definiert sein:

| **Datei** | **Variable** | **Bedeutung** |
|---|---|---|
| TERMCAP | gs | Zeichenfolge zum Einschalten des Grafikmodus. |
| | ge | Zeichenfolge zum Ausschalten des Grafikmodus. |
| | gb | Liste der ASCII-Zeichen, die für Grafiksteuerzeichen verwendet werden sollen. |
| TERMINFO | smacs | Zeichenfolge zum Einschalten des Grafikmodus. |
| | rmacs | Zeichenfolge zum Ausschalten des Grafikmodus. |
| | acsc | Liste der ASCII-Zeichen, die für Grafiksteuerzeichen verwendet werden sollen. |

Sie können in diesen Dateien weitere Grafikzeichen definieren. Mehr über das Ändern von TERMCAP bzw. TERMINFO siehe Kapitel 7.4 bzw. 7.5.

**Beispiel** zu Grafiksteuerzeichen im SCREEN-Abschnitt

Das Formatprogramm FORMAT05 enthält im SCREEN-Abschnitt Steuerzeichen, die einen Rahmen für das Format definieren:

```
screen
{
 \gp-------------------------------------------------------------q\g
 \g|\g                        FORMAT05                            \g|\g
 \g|\g                        --------                            \g|\g
 \g|-------------------------------------------------------------|\g
 \g|\g                                                            \g|\g
 \g|\g                                                            \g|\g
 \g|\g                                                            \g|\g
 \g|\g   Kundennummer         [f000       ]                       \g|\g
 \g|\g                                                            \g|\g
 \g|\g                                                            \g|\g
 \g|\g   Firma                [f001                 ]             \g|\g
 \g|\g                                                            \g|\g
 \g|\g   Vorname              [f002           ]                   \g|\g
 \g|\g   Nachname             [f003           ]                   \g|\g
 \g|\g                                                            \g|\g
 \gb-------------------------------------------------------------d\g
}
```

Daraus ergibt sich folgendes Format:

```
                        FORMAT05
                        ________

_____________________________________________________________


   Kundennummer         [       ]


   Firma                [                 ]

   Vorname              [           ]
   Nachname             [           ]
```

## TABLES-Abschnitt

Hier geben Sie alle Tabellen an, aus denen eine oder mehrere Spalten für das Formatprogramm entnommen werden.

```
TABLES
    [␣tab-alias = [datenbankname@system:][eigner.]]tabelle
    [␣...]
[END]
```

TABLES
: bezeichnet eine oder mehrere Tabellen. Wenn mehrere Tabellen verwendet werden, müssen sie durch Leerzeichen voneinander getrennt werden. Sie können maximal 16 Tabellen angeben. Wenn Sie INFORMIX-Netzprodukte verwenden (siehe Kapitel 1.1.2), können Sie auch Tabellen einer fremden Datenbank angeben (*datenbankname@system:*).

tab-alias
: definiert einen Alias-Namen für die angegebene Tabelle. Den Alias-Namen müssen Sie dann im Programm anstelle des Tabellennamens verwenden. Die Definition eines Alias-Namens ist Pflicht, wenn Sie *eigner* und/oder *datenbankname@system* angeben.

datenbankname
: Name der fremden Datenbank, die die gewünschte Tabelle enthält. Wenn Sie *datenbankname@system* angeben, müssen Sie auch *tab-alias* angeben.

system
: bezeichnet das INFORMIX-System. INFORMIX-SE: Name des Rechners, auf dem sich die fremde Datenbank befindet. INFORMIX-ONLINE: Servername, den der INFORMIX-ONLINE-Verwalter bei der Initialisierung vergeben hat.

eigner
: Name des Tabelleneigentümers. Wenn Sie *eigner* angeben, müssen Sie auch *tab-alias* angeben. Bei einer ANSI-Datenbank ist die Qualifikation mit dem Eigentümernamen Pflicht, wenn Sie Tabellen eines anderen Benutzers angeben.

tabelle

Name der Tabelle. Temporäre Tabellen sind nicht erlaubt. Die Angabe eines View ist nur erlaubt, wenn der View nur aus einer Tabelle erzeugt wurde und keine Mengenfunktion enthält. Ist die im DATABASE-Abschnitt angegebene Datenbank eine ANSI-Datenbank, dann dürfen externe Tabellen (Angaben *eigner* und/oder *datenbankname@system*) nur aus ANSI-Datenbanken stammen.

**Beispiel**

Das folgende Beispiel zeigt den TABLES-Abschnitt des Formatprogramms FORMAT05. Für die Tabelle *kunde* wird der Alias-Name *k* definiert und anschließend im ATTRIBUTES-Abschnitt anstelle des Tabellennamens verwendet.

```
tables
k = kunde
attributes
f000 = k.kunden_nr, reverse;
f001 = k.firma;
f002 = k.vorname;
f003 = k.nachname;
```

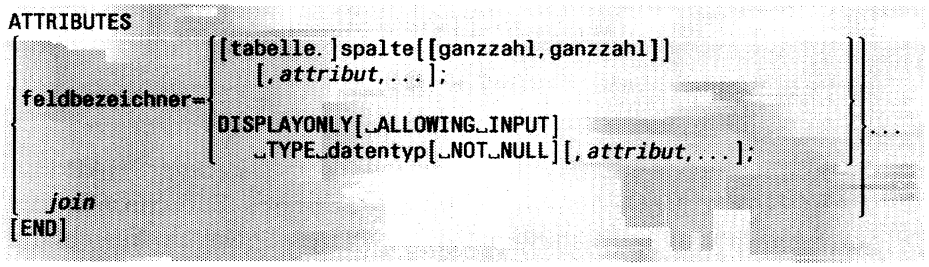
## ATTRIBUTES-Abschnitt

Jedes Bildschirmfeld, das Sie im SCREEN-Abschnitt mit einem Feldbezeichner gekennzeichnet haben, müssen Sie im ATTRIBUTES-Abschnitt wieder aufnehmen. Sie müessen dem Feldbezeichner

- eine Tabellenspalte oder
- ein DISPLAYONLY-Feld oder
- mehrere Tabellenspalten (Join)

zuweisen.

Darüberhinaus können Sie im ATTRIBUTES-Abschnitt Attribute für Bildschirm-Felder definieren (z.B. inverse Darstellung). Die Zuweisung von Attributen ist wahlfrei. Eine Beschreibung der möglichen Attribute finden Sie im anschließenden Abschnitt *Attribute.*




feldbezeichner
ist ein Feldbezeichner, der im SCREEN-Abschnitt verwendet wurde.

Die Reihenfolge, in der die Schreibmarke in PERFORM über das Format bewegt wird und Tabellen als aktuelle Tabelle aktiviert werden können, entspricht der Reihenfolge, in der die Feldbezeichner im ATTRIBUTES-Abschnitt untereinander aufgeführt werden. Durch die Anweisung NEXTFIELD können Sie von diesem standardmäßigen Vorgehen abweichen (siehe Abschnitt *Anweisungen für Kontrollblöcke* anschließend an den INSTRUCTIONS-Abschnitt).

[tabelle.]
Tabellenname bzw. Aliasname der Tabelle, wie im TABLES-Abschnitt definiert. Die Angabe ist nur erforderlich, wenn der angegebene Spaltenname nicht eindeutig ist, d.h. in mehr als einer Tabelle des Formatprogramms vorkommt.

spalte
ist der Name einer Tabellenspalte. Ist der Spaltenname in mehr als einer Tabelle des Formatprogramms enthalten, muß zur Eindeutigkeit der Tabellenname bzw. Aliasname mit angegeben werden (siehe *tabelle*).

*Beispiel*
(aus dem Formatprogramm *muster*, Anhang A.1):

```
k8      =ort;
```

Das Bildschirmfeld mit dem Feldbezeichner *k8* ist der Tabellenspalte *ort* der Tabelle *kunden* zugeordnet. In dieses Bildschirmfeld können Sie mit PERFORM

- Eingaben machen,
- sich Inhalte der Tabellenspalte *ort* anzeigen lassen und
- Werte zum Suchen aufnehmen.

[ganzzahl, ganzzahl]
Sie können sich Werte einer Tabellenspalte vom Datentyp CHAR und VARCHAR (INFORMIX-ONLINE) in mehreren Bildschirmfeldern anzeigen lassen. Dazu müssen Sie die CHAR-Spalte in Ausschnitte aufgliedern. Die eckigen Klammern [] müssen Sie angeben. Die Ausschnitte müssen dabei genau mit der Länge der im SCREEN-Abschnitt definierten Bildschirmfelder übereinstimmen.

*Beispiel*

```
f006 = kunde.name [1,20];
f007 = kunde.name [21,40];
```

Die Spalte *name* der Tabelle *kunde* ist auf zwei Bildschirmfelder aufgeteilt. Dabei enthält das Bildschirmfeld mit dem Feldbezeichner *f006* die Zeichen 1 - 20 der Werte der zugeordneten Tabellenspalte *kunde.name*. Entsprechend enthält das Bildschirmfeld des Bezeichners *f007* die Zeichen 21 bis 40 der gleichen Tabellenspalte. Die im SCREEN-Abschnitt definierten Bildschirmfelder der Feldbezeichner *f006* bzw. *f007* sind genau 20 Zeichen lang.

Im Standardformat werden Spalten, die länger als 50 Zeichen sind, automatisch auf mehrere Bildschirmfelder verteilt.

Eine andere Möglichkeit, CHAR-Werte mehrzeilig auszugeben, bietet das Attribut WORDWRAP.

*attribut*
finden Sie im anschließenden Abschnitt *Attribute* beschrieben.

DISPLAYONLY
zeigt, daß das angegebene Bildschirmfeld keiner Tabellenspalte zugeordnet ist. DISPLAYONLY-Felder dienen dazu, Ergebnisse einer Bedingung oder eines Ausdrucks anzuzeigen oder Werte aufzunehmen, die für die Berechnung einer Bedingung oder eines Ausdrucks benötigt werden. Die Werte eines DISPLAYONLY-Feldes werden nicht in der Datenbank gespeichert.

Ein DISPLAYONLY-Feld wird mit Werten versorgt entweder

- über den Benutzer (siehe ALLOWING INPUT) oder
- über eine Bedingung oder Berechnung im INSTRUCTIONS-Abschnitt.

Wenn Sie ein DISPLAYONLY-Feld ohne ALLOWING INPUT definieren, dann dürfen Sie nur die folgenden Attribute verwenden:

DEFAULT, DOWNSHIFT, FORMAT, QUERYCLEAR, REVERSE, RIGHT, UPSHIFT und ZEROFILL

Einschränkung für DISPLAYONLY-Felder des Datentyps BYTE (INFORMIX-ONLINE):
Als Inhalt erscheint grundsätzlich die Ausgabe <BYTE Wert>. Eingaben sind nicht möglich.

ALLOWING INPUT
erlaubt Eingaben eines Benutzers. Die Angabe wird beim Datentyp BYTE ignoriert.

Wenn Sie DISPLAYONLY-Felder mit ALLOWING INPUT definieren, sammelt PERFORM diese Bildschirmfelder in einer Tabelle namens *displaytable*. Es wird dabei keine Datenbanktabelle im eigentlichen Sinn erzeugt. PERFORM "tut" lediglich so, als ob *displaytable* eine Datenbanktabelle wäre. Sie können die Pseudotabelle *displaytable* verwenden,

- wenn Sie ein DISPLAYONLY-Feld mit PERFORM bearbeiten und
- wenn Sie einem DISPLAYONLY-Feld innerhalb eines Kontrollblocks Werte zuweisen (siehe Abschnitt *Kontrollblöcke* anschließend an den INSTRUCTIONS-Abschnitt).

Wenn eine Tabelle *displaytable* existiert, dann ist sie immer die letzte Tabelle in der Reihe der aktiven Tabellen, unabhängig davon, wo das DISPLAYONLY-Feld im ATTRIBUTES-Abschnitt auftaucht.

TYPE
ordnet dem DISPLAYONLY-Feld einen Datentyp zu.

datentyp
Name des Datentyps. Mit Ausnahme von SERIAL sind alle Datentypen erlaubt. Eine Beschreibung der Datentypen finden Sie im SQL-Handbuch [1].

Für den Datentyp CHAR dürfen Sie keine Länge definieren. Die Länge wird nämlich durch die Feldbegrenzer bestimmt.

Wenn Sie für die Typ DECIMAL oder MONEY die Genauigkeit festlegen, müssen Sie im SCREEN-Abschnitt ein entsprechend großes Feld bereitstellen.

Beim Datentyp BYTE wird die Angabe ALLOWING INPUT ignoriert.

NOT NULL
Sofern für das Feld auch ALLOWING INPUT definiert ist, wird der Benutzer gezwungen, Eingaben zu machen.

Wenn der DATABASE-Abschnitt die Definition WITHOUT NULL INPUT enthält und das DISPLAYONLY-Feld mit NOT NULL definiert wird, setzt PERFORM

- bei numerischen Feldern Nullen und
- bei CHAR-Feldern Leerzeichen

als Standardwerte ein.

*attribut*
finden Sie im anschließenden Abschnitt *Attribute* beschrieben.

*Beispiel* zu DISPLAYONLY-Feldern
(aus dem Formatprogramm *muster*, Anhang A.1):

```
d1    = displayonly type money;
d2    = displayonly type money;
```

Die Bildschirmfelder mit den Feldbezeichnern *d1* (Liefergebühr) und *d2* (Auftragsgesamtsumme) erhalten ihre Werte aufgrund von Berechnungen, die im INSTRUCTIONS-Abschnitt des Formatprogramms *muster* definiert sind. Diese Werte werden **nicht** in der Datenbank gespeichert.

*join*

Ein Format kann Daten aus mehreren Tabellen verwenden. Es kann dann ein Bildschirmfeld enthalten, das zwei oder mehr miteinander verbundenen Tabellenspalten zugeordnet ist. Diese Spalten stammen zwar aus verschiedenen Tabellen, haben aber einen gleichwertigen Datentyp. Sie verbinden Tabellenspalten, indem Sie ihnen denselben Feldbezeichner zuweisen.

Sie können zwei Arten von Joins definieren:

- einen einfachen, nicht überprüfenden Join und
- einen überprüfenden Join (durch das Setzen des Sterns *)

Ein einfacher Join bewirkt, daß Sie über die Join-Spalte einen (vorhandenen) verbundenen Satz angezeigt bekommen, ihn zum aktuellen Satz der Tabelle machen und somit auch korrigieren und löschen können. Außerdem können Sie eine MASTER/DETAIL-Beziehung definieren (siehe INSTRUCTIONS-Abschnitt).

Ein überprüfender Join gibt Ihnen durch das Setzen des Sterns * die Möglichkeit, Werte nur noch in Abhängigkeit zu verbundenen Tabellen neuaufzunehmen, zu löschen oder zu korrigieren.

```
join::=
feldbezeichner=[*][tabelle1.]spalte1[,attribut,...][;]...
              =[*][tabelle2.]spalte2[,attribut,...];
```

feldbezeichner

ist einer der Feldbezeichner, der im SCREEN-Abschnitt verwendet wurde.

tabelle1.spalte1, tabelle2.spalte2

sind die Tabellenspalten, die Sie durch einen Join miteinander verbinden.

Die Tabellenspalten müssen den gleichen bzw. gleichwertigen Datentyp haben. Hierbei gelten SERIAL-, INTEGER- und DATE-Spalten als gleich. Sie dürfen jedoch keine SERIAL-Spalten miteinander verbinden. Eine SERIAL-Spalte können Sie nur mit einer Tabellenspalte des Datentyps INTEGER verbinden. Tabellenspalten vom Datentyp CHAR müssen gleich lang sein.

Die Tabellenspalten müssen nicht indiziert sein. Eine Indizierung erhöht jedoch die Ablaufgeschwindigkeit von Datenbankabfragen.

Sie können Tabellenspalten aus maximal 16 Tabellen miteinander verbinden.

*Beispiel* zu Join
(aus dem Formatprogramm *muster*, Anhang A.1):

```
au11 = * auftrag.auftrags_nr
         = posten.auftrags_nr;
```

Der Feldbezeichner *au11* verbindet die Spalte *auftrags_nr* der Tabelle *auftrag* mit der Spalte *auftrags_nr* der Tabelle *posten*. Der Stern vor der Angabe *auftrag.auftrags_nr* zeigt an, daß es sich um einen überprüfenden Join handelt.

*attribut*

Wenn eine oder mehrere Attribute bei einem Join nur dann wirksam sein sollen, wenn sich der Benutzer in der jeweils aktiven Tabelle befindet,

- geben Sie das Attribut nur bei derjenigen Join-Spalte an, bei der Sie das Auftreten des Attributs wünschen
- trennen die einzelnen Join-Spalten durch Semikolon (;) ab und
- schließen den gesamten Join mit Semikolon (;) ab

Wenn ein oder mehrere Attribute bei mehreren Tabellen-Spalten wirksam sein sollen, gleichgültig welche dieser Tabellen aktiv ist:

- geben Sie das Attribut bei der letzten Join-Spalte des Bündels von Spalten an, für die dieses Attribut gelten soll
- trennen die einzelnen Bündel von Spalten durch Semikolon (;) ab und
- schließen den gesamten Join mit Semikolon (;) ab

Wenn eine oder mehrere Attribute bei allen Join-Spalten wirksam sein sollen, gleichgültig welche Tabelle aktiv ist:

- geben Sie das Attribut bei der letzten Tabellenspalte des Joins an und
- schließen den gesamten Join mit Semikolon (;) ab

*Hinweis:*
Das Attribut REVERSE stellt eine Ausnahme dar. REVERSE ist immer wirksam, gleichgültig welche Tabelle aktiv ist.

Eine Erläuterung zu den möglichen Attributen finden Sie im anschließenden Abschnitt *Attribute.*

*Beispiel* zu attribut
(aus dem Formatprogramm *muster*, Anhang A.1):

```
p13   = posten.artikeln_nr;
        = * artikel.artikeln_nr, noentry, noupdate, queryclear;
```

Der Feldbezeichner *p13* verbindet die Spalten *posten.artikel_nr* und *artikel.artikel_nr* miteinander. Es handelt sich um einen überprüfenden Join. Dabei ist die Spalte *artikel.artikel_nr* dominant.

Die Attribute NOENTRY, NOUPDATE und QUERYCLEAR (siehe anschließender Abschnitt *Attribute*) sind nur wirksam, wenn die Tabelle *artikel* aktiv ist.

* Mit dem Setzen des Sterns * definieren Sie einen überprüfenden Join. In einem überprüfenden Join gibt es immer eine dominante Spalte bzw. eine dominante Tabelle. Die dominante Spalte ist immer die Spalte, die auf den Stern * folgt. Sie dürfen innerhalb eines Joins nur eine Spalte zur dominanten Spalte machen.

Durch einen überprüfenden Join wird getestet, ob ein Wert, der in eine Join-Spalte eingetragen werden soll, einem Wert in der verbundenen Tabellenspalte, der dominanten Spalte, entspricht. Falls der Wert in der dominanten Tabelle nicht vorhanden ist, kann der Wert nicht eingetragen werden. Einen Satz aus der dominanten Tabelle kann ein Benutzer nur korrigieren oder löschen, wenn er zuvor die entsprechenden Sätze in verbundenen Tabellen gelöscht oder korrigiert hat.

*Beispiel* zu einem überprüfenden Join
(aus dem Formatprogramm *muster*, Anhang A.1):

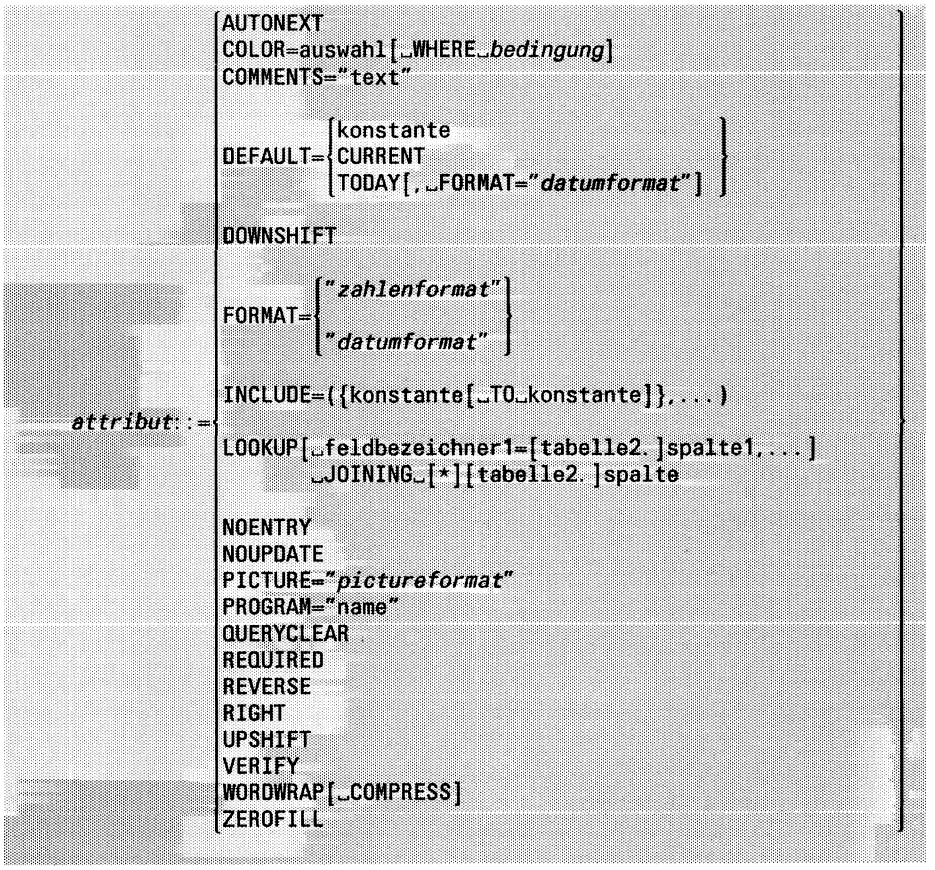
```
k1    = * kunde.kunden_nr
          = auftrag.kunden_nr;
```

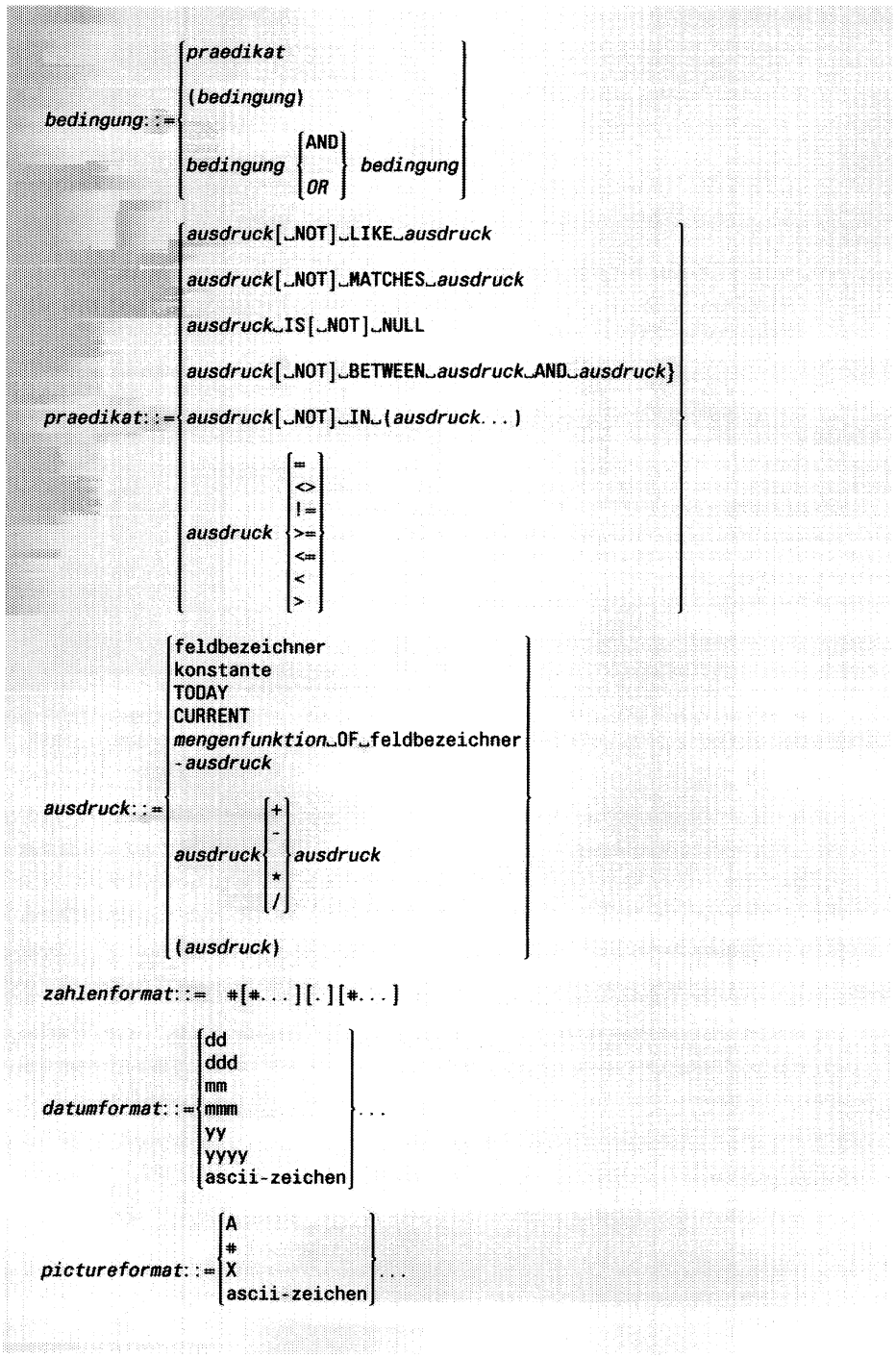
In diesem überprüfenden Join ist die Spalte *kunden.kunden_nr* dominant. Wenn die Tabelle *auftrag* aktiv ist, darf ein Benutzer in das Bildschirmfeld mit dem Feldbezeichner *k1* keinen Wert eintragen, der nicht bereits in der Spalte *kunden_nr* der Tabelle *kunden* existiert. Auf diese Weise wird sichergestellt, daß zwischen den Kundennummern der Tabelle *kunde* und der Tabelle *auftrag* keine Inkonsistenzen auftreten.

### Attribute

Über *attribut* weisen Sie einem Bildschirmfeld Eigenschaften zu, wie z.B. inverse Darstellung des Bildschirmfeldes. Die Zuweisung von Attributen ist frei wählbar. Sie können jedoch die Bearbeitung eines Formats erleichtern, indem Sie Attribute für ein Feld angeben.

Die Attribute sind wirksam, wenn Sie sich in der aktuellen Tabelle befinden. Eine Ausnahme stellt der Join (siehe ATTRIBUTES-Abschnitt, *join*) und das Attribut LOOKUP dar.




bedingung::=
praedikat
(bedingung)
bedingung AND OR bedingung
praedikat::=
ausdruck[ NOT] LIKE ausdruck
ausdruck[ NOT] MATCHES ausdruck
ausdruck IS[ NOT] NULL
ausdruck[ NOT] BETWEEN ausdruck AND ausdruck}
ausdruck[ NOT] IN (ausdruck...)
ausdruck = <> != >= <= < >
ausdruck::=
feldbezeichner
konstante
TODAY
CURRENT
mengenfunktion OF feldbezeichner
-ausdruck
ausdruck + - * / ausdruck
(ausdruck)
zahlenformat::= #[#...][.][#...]
datumformat::=
dd
ddd
mm
mmm
yy
yyyy
ascii-zeichen
...
pictureformat::=
A
#
X
ascii-zeichen
...

AUTONEXT

bewegt die Schreibmarke in PERFORM automatisch in das nächste Bildschirmfeld, sobald das aktuelle Feld gefüllt ist.

Ein Bildschirmfeld ist gefüllt,

- wenn der Benutzer bei der Eingabe entweder am letzten Zeichen der definierten Feldlänge angekommen ist, oder
- wenn für ein Feld irgendein Standard definiert wurde, z.B. über das Attribut FORMAT oder INCLUDE und dieser Standard mit der Eingabe erfüllt wurde

AUTONEXT ist besonders sinnvoll bei Tabellen-Spalten des Datentyps CHAR, die Sie in mehrere Bildschirmfelder aufgeteilt haben (siehe SCREEN-Abschnitt, [tabelle.]spalte).

*Beispiel* zu AUTONEXT
(aus dem Formatprogramm *muster*, Anhang A.1):

```
k9    = bundesland, upshift, autonext,
        include = ("HA","BY","BW","BE"),
        default = "BY" ,
        comments = "Eingabe erlaubt:'HA','BY','BW'- Standartwert: 'BY';
```

Der Wert des Bildschirmfeldes *Bundesland* ist durch das Attribut DEFAULT voreingestellt. Vom Benutzer kann dieser voreingestellte Wert aber mit einer der Abkürzungen, die bei INCLUDE definiert sind, überschrieben werden. Wenn der Wert überschrieben wird, geht die Schreibmarke aufgrund des Attributs AUTONEXT zum nächsten Bildschirmfeld, ohne daß der Benutzer die Taste [↵] drücken muß. Alle Eingaben in das Bildschirmfeld *Bundesland* werden in Großbuchstaben dargestellt (siehe UPSHIFT).

COLOR

definiert Eigenschaften zur Gestaltung des Feldinhalts, wie Farbe, Blinken, Unterstreichung, Invertierung, linksbündiges Ausrichten. Sie können dabei eine Bedingung angeben, unter der die gewählte Eigenschaft verwendet werden soll.

auswahl

definiert die Eigenschaft. Folgende Angaben sind möglich:

| **Eigenschaft** | **Darstellung des Feldinhalts** |
|---|---|
| WHITE<br>YELLOW<br>MAGENTA<br>RED<br>CYAN<br>GREEN<br>BLUE<br>BLACK | entsprechende Farbe |
| BLINK | blinkend |
| UNDERLINE | unterstrichen |
| REVERSE | invers (helle Schrift auf dunklem Grund oder umgekehrt) |
| LEFT | linksbündig ausgerichtet |

*auswahl* läßt folgende Kombinationen zu:

- 1 Farbe
- BLINK, UNDERLINE, REVERSE und LEFT in beliebiger Kombination
- 1 Farbe, beliebig kombiniert mit BLINK, UNDERLINE, REVERSE und LEFT

WHERE *bedingung*

gibt an, daß die in *auswahl* definierten Eigenschaften nur unter der angegebenen Bedingung verwendet werden sollen. Die Beschreibung sämtlicher Elemente für *bedingung* finden Sie im SQL-Handbuch [1], Kapitel 5. Sie dürfen nur die Elemente verwenden, die in der Syntaxübersicht von *attribute* angegeben sind.

Fehlt die Angabe einer Bedingung, wird die in *auswahl* definierte Eigenschaft für alle Feldinhalte verwendet.

*Beispiel* zu COLOR

Der Feldinhalt des Bildschirmfeldes *f000* soll in roter Farbe ausgegeben werden:

```
f000 = kunde. kunden_nr, color = red
```

Nur der Feldinhalt "HRO" des Bildschirmfeldes *f002* soll in roter Farbe erscheinen:

```
f002 = hersteller. hersteller_code,
       color = red where f002 = "HRO"
```

Alle Postleitzahlen ≠ 8000 im Feld *f003* sollen blinkend und in roter Farbe angezeigt werden:

```
f003 = kunde. plz color = red blink where f003 ! = 8000
```

COMMENTS

schreibt beim Neuaufnehmen, Suchen und Korrigieren einen Kommentar in die Kommentarzeile am unteren Bildschirmrand des Formats.

"text"

Hier geben Sie den gewünschten Kommentar ein. Er muß in Anführungszeichen eingeschlossen werden. Der Kommentar darf nur eine Bildschirmzeile umfassen.

*Beispiel* zu COMMENTS
(aus dem Formatprogramm *muster*, Anhang A.1):

```
k2    = vorname,
        comments = "Bei Bedarf Vornamen eingeben";
```

Während des Neuaufnehmens, Korrigierens und Suchens im Bildschirmfeld *vorname* erscheint am unteren Bildschirmrand der Kommentar *Bei Bedarf Vornamen eingeben.*

DEFAULT
weist einem Bildschirmfeld einen Standardwert zu, den PERFORM beim Neuaufnehmen automatisch einträgt. Der Standardwert ist überschreibbar.

DEFAULT darf nicht bei den INFORMIX-ONLINE-Datentypen TEXT und BYTE angegeben werden.

Ist DEFAULT angegeben, so wird das Attribut REQUIRED ignoriert. Fehlt die Angabe DEFAULT, wird der Standardwert aus WITHOUT NULL INPUT (DATABASE-Abschnitt) zugewiesen. Fehlt auch WITHOUT NULL INPUT, wird als Standardwert ein NULL-Wert zugewiesen.

konstante
ist der Standardwert, den Sie über DEFAULT einem Bildschirmfeld zuweisen.

Wenn eine *konstante* vom Datentyp DATE, CHAR oder VARCHAR (INFORMIX-ONLINE) Leer- oder Sonderzeichen enthält, müssen Sie *konstante* in Anführungszeichen einschliessen. Das Setzen von Anführungszeichen ist in allen anderen Fällen bei den Datentypen CHAR und VARCHAR frei wählbar.

*Beispiel* zu DEFAULT
(aus dem Formatprogramm *muster*, Anhang A.1):

```
k9    = bundesland, upshift, autonext,
        include = ("HA","BY","BW","BE"),
        default = "BY" ,
        comments = "Eingabe erlaubt:'HA','BY','BW'- Standartwert: 'BY';
```

Der Wert des Bildschirmfeldes *Bundesland* ist durch das Attribut DEFAULT voreingestellt. Vom Benutzer kann dieser voreingestellte Wert aber mit einer der Abkürzungen, die bei INCLUDE definiert sind, überschrieben werden. Wenn der Wert überschrieben wird, geht die Schreibmarke aufgrund des Attributs AUTONEXT zum nächsten Bildschirmfeld, ohne daß der Benutzer die Taste ↵ drücken muß. Alle Eingaben in das Bildschirmfeld *Bundesland* werden in Großbuchstaben dargestellt (siehe UPSHIFT).

CURRENT
vereinbart für den Datentyp DATETIME als Standardwert das aktuelle Datum und die Uhrzeit (maximale Genauigkeit: 1/100 Sekunde). CURRENT darf nur beim Datentyp DATETIME angegeben werden.

TODAY
vereinbart für den Datentyp DATE als Standardwert das aktuelle Datum. Die Angabe ist nur beim Datentyp DATE erlaubt.

"datumformat"
ist nur erlaubt, wenn das Bildschirmfeld einer DATE-Spalte zugeordnet ist. Die Wirkung von *datumformat* ist beim Attribut FORMAT beschrieben.

*Beispiel* zu TODAY
(aus dem Formatprogramm *muster*, Anhang A.1):

```
au12 = Auftragsdatum, default = today, format = "dd. mm. yy";
```

Der Wert des Bildschirmfeldes mit dem Feldbezeichner *au12* (Auftragsdatum) ist mit dem aktuellen Datum voreingestellt. Das Datum wird mit jeweils 2 Ziffern für Tag, Monat und Jahr dargestellt.

DOWNSHIFT
wandelt bei folgenden Datentypen Großbuchstaben in Kleinbuchstaben um: CHAR, VARCHAR, TEXT. Die Angabe wird bei allen anderen Datentypen ignoriert.

Da Groß- und Kleinbuchstaben unterschiedliche ASCII-Werte haben, können Sie durch Festlegung auf entweder nur Großbuchstaben oder nur Kleinbuchstaben das Abfragen einer Datenbank vereinfachen.

FORMAT
formatiert Zahlen oder Datumsangaben. FORMAT darf nicht für die Datentypen DATETIME und INTERVAL verwendet werden.

"zahlenformat"
beschreibt die Lage des Kommas bei Bildschirm-Feldern, die Spalten des Datentyps DECIMAL, SMALLFLOAT oder FLOAT zugeordnet sind. Ein "zahlenformat" besteht aus den Zeichen #, das die einzelnen Stellen darstellt. Wahlweise darf maximal ein Punkt '.' für die Dezimalstelle angegeben werden.

Ist der anzuzeigende Wert kürzer als *zahlenformat*, richtet PERFORM den Wert im Bildschirmfeld nach rechts aus und füllt von links mit Leerzeichen auf.

Ist der anzuzeigende Wert länger als *zahlenformat*, rundet PERFORM ab und gibt eine Warnung aus.

"datumformat"
ist nur erlaubt, wenn das Bildschirmfeld einer DATE-Spalte zugeordnet ist.

Es gibt für *datumformat* folgende Möglichkeiten:

| **Formatteil** | **Anzeige** |
|---|---|
| mm | erzeugt eine zweistellige Darstellung des Monats. |
| mmm | erzeugt eine drei Buchstaben lange Abkürzung für den Monat. z.B. Jan, Feb, usw. |
| dd | erzeugt eine zweistellige Darstellung des Tagesdatums. |
| ddd | erzeugt eine drei Buchstaben lange Abkürzung für den Tagesnamen. |
| yy | erzeugt eine zweistellige Darstellung der Jahreszahl. |
| yyyy | erzeugt eine vierstellige Darstellung der Jahreszahl. |
| beliebiges ASCII-Zeichen | nimmt nur sich selbst auf und dient als Begrenzung zwischen *Tag*, *Monat* und *Jahr*. |

*Beispiel* zu datumformat
(aus dem Formatprogramm *muster*, Anhang A.1):

```
au22 = zahldatum, format = "dd. mm. yy";
```

Die Eingabe in das Bildschirmfeld *Zahldatum* wird z.B. in der Form '23.06.90' dargestellt. Dabei ist es gleichgültig, ob z.B. '23,06,90' oder '23,6,90' eingegeben wurde.

INCLUDE

definiert für ein Feld zulässige Werte. PERFORM akzeptiert Eingabewerte nur, wenn sie zu den definierten Werten gehören.

INCLUDE wird bei einem Bildschirmfeld vom Datentyp SERIAL ignoriert.

Wenn Sie für ein Bildschirmfeld sowohl INCLUDE als auch DEFAULT definieren, so muß der Wert von DEFAULT bei INCLUDE angegeben sein.

Felder mit dem *attribut* INCLUDE zwingen den Benutzer zur Eingabe eines bei INCLUDE definierten Wertes.

Den gültigen Wertebereich sollten Sie, um die Arbeit mit dem Format zu erleichtern, mit COMMENTS als Kommentar zum Feld ausgeben.

konstante

läßt Wertangaben in folgender Form zu:

- als Aufzählung von Einzelwerten, die Sie dann durch Kommas trennen (konstante,...)
- Bereichsangabe, wobei Anfangs- und Endwert durch das Schlüsselwort TO verbunden werden (konstante TO konstante,...)
- Wert NULL, der die Angabe eines NULL-Wertes zuläßt.

Bei der Definition eines Wertebereichs über Anfangs- und Endwerte muß der kleinere Wert zuerst stehen. Beachten Sie, daß in einem numerischen Feld beispielsweise der Eingabebereich (5 TO 10) korrekt ist. Dies gilt aber nicht für ein CHAR-Feld, weil hier die Stelligkeit der Ziffern gar nicht berücksichtigt werden kann. Das Zeichen 5 stellt in diesem Fall ein ASCII-Zeichen dar. Im ASCII-Zeichensatz erscheint 10 vor 5. Dadurch ist das Zeichen 10 kleiner als das Zeichen 5. Eine Auflistung des ASCII-Zeichensatzes finden Sie im Anhang A.3.

Werte, die Leerzeichen, Kommas oder Sonderzeichen enthalten, müssen in Anführungszeichen eingeschlossen werden.

*Beispiel* zu INCLUDE
(aus dem Formatprogramm *muster*, Anhang A.1):

```
p18   = posten.Menge, include = (1 to 50),
        comments = "Menge zwischen 1 und 50" ;
```

In das Bildschirmfeld *Menge* dürfen nur Werte zwischen 1 und 50 eingegeben werden. Gleichzeitig erscheint am unteren Bildschirmrand als Hilfe für den Benutzer der Kommentar "Menge zwischen 1 und 50".

LOOKUP
zeigt Spalten einer anderen Tabelle an, während der Benutzer mit der aktuellen Tabelle arbeitet.

feldbezeichner1 = tabelle2.spalte1
macht das Bildschirmfeld mit dem *feldbezeichner1* zum LOOKUP-Feld. Wenn Sie für *tabelle2* einen Aliasnamen vereinbart haben (TABLES-Abschnitt), dann müssen sie hier den Alias-Namen verwenden. Im ATTRIBUTES-Abschnitt schreiben Sie den *feldbezeichner1* nur an diese Stelle. In diesem Feld, dem LOOKUP-Feld, zeigt PERFORM die Werte der Tabellenspalte *tabelle2.spalte1* an:

- wenn die verbundene Tabelle aktiv ist und
- wenn PERFORM verbundene Sätze zwischen den verbundenen Tabellen findet

In ein LOOKUP-Feld lassen sich erst dann Werte eingeben, wenn die Tabelle aktiv ist, deren Spalte dem LOOKUP-Feld zugeordnet ist.

Die Angabe eines Spaltenausschnitts (in der Form *spalte[ganzzahl,ganzzahl]*, siehe ATTRIBUTES-Abschnitt) ist bei LOOKUP-Feldern nicht erlaubt.

JOINING
muß vor *tabelle2.spalte* stehen.

tabelle2.spalte
bezeichnet die Tabellenspalte, das Sie über JOINING mit der Tabellenspalte von *feldbezeichner* verbinden (siehe Angabe *join* im ATTRIBUTES-Abschnitt). Die Angabe eines Spaltenausschnitts (in der Form *spalte[ganzzahl,ganzzahl]*, siehe ATTRIBUTES-Abschnitt) ist nicht erlaubt.

Wenn Sie für *tabelle2* einen Aliasnamen vereinbart haben (TABLES-Abschnitt), dann müssen sie hier den Alias-Namen verwenden.

Die beiden verbundenen Tabellenspalten müssen nicht indiziert sein. Eine Indizierung erhöht jedoch die Ablaufgeschwindigkeit.

* Sie setzen wahlweise den Stern, um Eingaben zu überprüfen. PERFORM akzeptiert einen Wert für das Feld von *feldbezeichner* nur dann, wenn der gleiche Wert bereits in der dominanten Spalte *tabelle2.spalte* vorhanden ist. Sie können dadurch verhindern, daß der Benutzer Werte eingibt, die in der dominanten Tabelle nicht vorhanden sind.

*Beispiel* zu LOOKUP
(aus dem Formatprogramm *muster*, Anhang A.1):

```
p16  =  posten.herstellercode,
            lookup m17 = hersteller.herstellername
            joining *hersteller.herstellercode, upshift;
        = * artikel.herstellercode, noentry, noupdate,
            upshift, autonext, queryclear;
```

Im Bildschirmfeld mit dem Feldbezeichner *p16* (Hersteller) werden die Tabellen *posten* und *artikel* durch einen überprüfenden Join miteinander verbunden. D.h., wenn die Tabelle *posten* aktiv ist, können in das Bildschirmfeld mit dem Feldbezeichner *p16* (Hersteller) nur solche Werte eingetragen werden, die bereits in der Tabelle *artikel* vorhanden sind. Zusätzlich beinhaltet das obige Beispiel einen überprüfenden LOOKUP-Join. D.h., wenn die Tabelle *posten* aktiv ist, können in das Bildschirmfeld mit dem Feldbezeichner *p16* (Hersteller) zusätzlich nur solche Werte eingetragen werden, die in der Spalte *herstellercode* der Tabelle *hersteller* gefunden werden können. Das Bildschirmfeld mit dem Feldbezeichner *m17* (*herstellername*) ist ein LOOKUP-Feld. In ihm wird der zum verbundenen Satz gehörende Wert der Spalte *hersteller.herstellername* angezeigt.

NOENTRY
verhindert das Neuaufnehmen von Werten. Werte werden weder über den Benutzer, noch über die Anweisung LET aufgenommen.

Bei SERIAL-Feldern ist NOENTRY unnötig.

NOENTRY hindert nicht daran, während des Korrigierens Werte zu ändern.

*Beispiel* zu NOENTRY
(aus dem Formatprogramm *muster*, Anhang A.1):

```
ar14 = artikel. bezeichnung, noentry, noupdate;
```

Wenn die Tabelle *artikel* aktiv ist, kann in das Bildschirmfeld *bezeichnung* kein Wert eingetragen werden. Das Attribut NOUPDATE verhindert zusätzlich, daß Werte beim Korrigieren eingetragen werden können.
Die Tabellenspalte *bezeichnung* erhält seine Werte entweder über eine INSERT-Anweisung (siehe SQL-Handbuch [1]) oder über ein anderes Format.

NOUPDATE
verhindert das Korrigieren. Werte können weder über den Benutzer, noch über die Anweisung LET geändert werden. NOUPDATE hindert nicht daran, während des Neuaufnehmens Werte zu ändern.

*Beispiel* zu NOUPDATE
(aus dem Formatprogramm *muster*, Anhang A.1):

```
ar15 = artikel. preis, noentry, noupdate;
```

Wenn die Tabelle *artikel* aktiv ist, kann das Bildschirmfeld mit dem Feldbezeichner *ar15* (Preis) nicht korrigiert werden. Das Attribut NOENTRY verhindert zusätzlich, daß Werte beim Neuaufnehmen eingetragen werden können.

PICTURE
definiert "pictureformat" für einen Eintrag in ein Bildschirmfeld, das einer Spalte vom Datentyp CHAR zugeordnet ist.

"pictureformat"
ist eine Zeichenkette, die das gewünschte Eingabemuster enthält.

Die Zeichenkette kann aus A, #, X oder einem beliebigen ASCII-Zeichen bestehen und muß in Anführungszeichen eingeschlossen werden.

| Symbol | Bedeutung |
|---|---|
| A | nimmt nur Buchstaben auf |
| # | nimmt nur Ziffern auf |
| X | nimmt jedes Zeichen auf |

beliebiges ASCII-Zeichen nimmt nur sich selbst auf

Die Zeichenkette und das im SCREEN-Abschnitt definierte Bildschirmfeld müssen gleich lang sein.

PERFORM erinnert den Benutzer an *pictureformat*, indem es die beliebigen ASCII-Zeichen im Bildschirmfeld anzeigt und für alle anderen Symbole Leerzeichen setzt.

Wenn Sie versuchen, eine Eingabe zu machen, die nicht mit der definierten Zeichenkette übereinstimmt, ertönt ein akustisches Signal.

Um dem Benutzer die Arbeit zu erleichtern, sollten Sie mit COMMENTS ein Beispiel zu *pictureformat* mitausgeben.

*Beispiel* zu PICTURE
(aus dem Formatprogramm *muster*, Anhang A.1):

```
k10      = telefon, picture = "#####/########";
```

Bevor irgendwelche PERFORM-Operationen durchgeführt werden, sieht das Bildschirmfeld *Telefon* folgendermaßen aus:

```
[      /           ]
```

Der Benutzer kann Werte nur in diese Vorgabe eintragen.

PROGRAM = "name"

Dieses Attribut verwenden Sie nur für die INFORMIX-ONLINE-Datentypen TEXT und BYTE. Über PROGRAM vereinbaren Sie ein Programm, mit dem Sie den Inhalt Ihres TEXT- oder BYTE-Feldes bearbeiten wollen.

*Ablauf:*

Bei Ablauf des Formats (PERFORM), nach Auswahl der gewünschten Funktion (*Blob*, *Neuaufnehmen* oder *Korrigieren*), geben Sie an der Anfangsposition des Bildschirmfeldes ein Ausrufezeichen "!" ein. INFORMIX kopiert dann den zugeordneten TEXT- oder BYTE-Wert in eine temporäre Datei auf Platte. Anschließend startet INFORMIX das entsprechende Programm für diese Datei. Nach Programmende gibt INFORMIX wieder den Format-Bildschirm aus. Den Feldinhalt speichern Sie dann wie gewohnt über PERFORM in die Tabelle ein.

Fehlt die Angabe PROGRAM, so gilt für den

- Datentyp TEXT: Der Editor, der in der Umgebungsvariable DBEDIT definiert ist, wird aktiviert. Ist DBEDIT nicht gesetzt, so gilt der Standardeditor des Betriebssystems. (Umgebungsvariable DBEDIT siehe 7.2)

- Datentyp BYTE: Der Inhalt kann nicht bearbeitet werden.

"name"
bezeichnet das Programm, mit dem der Feldinhalt bearbeitet werden soll. Mit *name* geben Sie entweder das Betriebssystemkommando an, das das gewünschte Programm aufruft oder eine Kommandodatei, in der die entsprechenden Aufrufe stehen. *name* muß nicht aus einem einzelnen Wert bestehen.

*Beispiel* zu PROGRAM

Das Bildschirmfeld *buch* soll bei Formatablauf mit dem Editor *maxed* bearbeitet werden.

```
f10=buch, program="maxed";
```

QUERYCLEAR
löscht den Inhalt eines Bildschirmfeldes, sobald Sie die PERFORM-Funktion *Suchen* aufrufen. Dieses Attribut ist nur für Bildschirmfelder sinnvoll, in denen Tabellen durch Join verbunden sind. PERFORM löscht nämlich bei Aufruf der Funktion *Suchen* mit Ausnahme von JOIN- und DISPLAYONLY-Feldern sämtliche Bildschirmfelder.
Für DISPLAYONLY-Felder kann QUERYCLEAR jedoch nicht verwendet werden. Dort ist das Löschen nur über Angaben im INSTRUCTIONS-Abschnitt möglich.

*Beispiel* zu QUERYCLEAR
(aus dem Formatprogramm *muster*, Anhang A.1):

```
p13  = posten.artikel_r;
       = * artikel.artikel_r, noentry, noupdate, queryclear;
```

Die Tabellen *posten* und *artikel* sind über die Artikelnummer durch einen überprüfenden Join verbunden. Wenn die Tabelle *artikel* aktiv ist und die Menüfunktion *Suchen* durchgeführt wird, wird der Inhalt des Feldes *Artikelnummer* gelöscht. Wenn jedoch beim Suchen die Tabelle *posten* aktiv ist, wird der Inhalt des Feldes *Artikelnummer* nicht gelöscht.

REQUIRED
zwingt beim Neuaufnehmen zu einer Eingabe in das Bildschirmfeld.

Während des Korrigierens hat REQUIRED keine Wirkung.

Wenn Sie einem Feld sowohl REQUIRED als auch DEFAULT zuweisen, wird REQUIRED ignoriert.

*Beispiel* zu REQUIRED
(aus dem Formatprogramm *muster*, Anhang A.1):

```
au20 = fremd_nr, required,
       comments = "Falls keine fremde Nummer, Name des Anrufers" ;
```

PERFORM erwartet beim Bildschirmfeld mit dem Feldbezeichner *au20* (Fremde Nummer) einen Eintrag.

REVERSE
stellt ein Bildschirmfeld invers, d.h. mit dunkler Schrift auf hellem Hintergrund bzw. heller Schrift auf dunklem Hintergrund, dar.

RIGHT
richtet Feldinhalte beim Neuaufnehmen und Korrigieren rechtsbündig aus.

In einem Standardformat ist RIGHT bei numerischen Feldern automatisch angegeben.

Beim Suchen in einem rechtsbündig ausgerichteten Feld des Datentyps CHAR muß dem Suchwert ein '*' vorausgehen, um Leerzeichen am Anfang des Ausdrucks zu ignorieren.

UPSHIFT

wandelt bei folgenden Datentypen Kleinbuchstaben in Großbuchstaben um: CHAR, VARCHAR, TEXT. Die Angabe wird bei allen anderen Datentypen ignoriert.

Da Groß- und Kleinbuchstaben unterschiedliche ASCII-Werte haben, können Sie durch Festlegung auf entweder nur Großbuchstaben oder nur Kleinbuchstaben das Abfragen einer Datenbank vereinfachen.

*Beispiel* zu UPSHIFT
(aus dem Formatprogramm *muster*, Anhang A.1):

```
k9    = bundesland, upshift, autonext,
        include = ("HA","BY","BW","BE"),
        default = "BY" ,
        comments = "Eingabe erlaubt:'HA','BY','BW'- Standartwert: 'BY';
```

Der Wert des Bildschirmfeldes *Bundesland* ist aufgrund des Attributs DEFAULT voreingestellt. Wenn der Benutzer jedoch diesen Wert mit einer der unter INCLUDE definierten Abkürzungen überschreiben will, werden alle Eingaben aufgrund des Attributs UPSHIFT automatisch in Großbuchstaben dargestellt.

VERIFY

fordert dazu auf, eine Eingabe ein zweites Mal zu wiederholen, um Eingabefehler auszuschließen. Als Kommentar am unteren Bildschirmrand erscheint: *Eingabe zur Kontrolle wiederholen.*

PERFORM akzeptiert die zweite Eingabe nur dann, wenn sie mit der ersten Eingabe genau identisch ist.

*Beispiel* zu VERIFY

Eingaben in das Bildschirmfeld *gehalt* sollen zur Kontrolle wiederholt werden.

```
f11 = Gehalt, verify;
```

WORDWRAP

Mit WORDWRAP können Sie einen Spaltenwert auf mehrere Bildschirmfelder verteilen, um z. B. für ein langes CHAR-Feld eine mehrzeilig Ausgabe zu ermöglichen. Dabei definieren Sie im SCREEN-Abschnitt mehrere Bildschirmfelder unter dem Namen des Feldbezeichners, für den Sie WORDWRAP vereinbaren. WORDWRAP darf nur für die Datentypen CHAR, VARCHAR und TEXT verwendet werden. (VARCHAR und TEXT stehen nur dem INFORMIX-ONLINE-Anwender zur Verfügung.)

Bei Ablauf des Formates (PERFORM) benutzen Sie für die Bearbeitung eines solchen Bildschirmfeldes den *Multiline-Editor.* Dieser bietet abweichend von den allgemeinen Bedienungsfunktionen von PERFORM zusätzliche Funktionen zur komfortablen Bearbeitung des Feldinhalts. Die Beschreibung des *Multiline-Editors* finden Sie in Kapitel 4.4.4.

Für den Datentyp TEXT gilt folgende Einschränkung: Der Inhalt eines TEXT-Feldes läßt sich nicht über den Multiline-Editor bearbeiten. WORDWRAP dient hier nur zur Anzeige des Inhalts. Zur Bearbeitung des Inhalts können Sie über das Attribut PROGRAM einen Editor vereinbaren.

*Größe des Bildschirmfeldes bei WORDWRAP:*

Da der *Multiline Editor* automatisch Leerzeichen in das Feld schreibt, sollten Sie dafür zusätzlichen Platz im Bildschirmfeld berücksichtigen. Als Richtwert gilt: die Anzahl an Editor-Leerzeichen je Feldteil beträgt:

1/2 * Summe aller Wörter dieses Feldteils

Für CHAR- und VARCHAR-Werte gibt es noch eine andere Möglichkeit, den Inhalt auf mehrere Bildschirmfelder zu verteilen (siehe dazu unter ATTRIBUTES/Tabellenspalte die Angabe [ganzzahl,ganzzahl]). Dort haben Sie jedoch nicht die Möglichkeit, den Feldinhalt mit dem *Multiline-Editor* zu bearbeiten.

[COMPRESS]
löscht alle Leerzeichen '␣', die Sie nicht explizit eingegeben haben. Solche Leerzeichen entstehen z. B. am Ende jedes Feldteils, wenn der Multiline-Editor einen Zeilenumbruch durchführt.
Fehlt die Angabe, so werden die Leerzeichen entsprechend übernommen.

*Beispiel* zu WORDWRAP

Das folgende Formatprogramm zeigt die Definition eines WORDWRAP-Feldes in SCREEN- und ATTRIBUTES-Abschnitt.

```
database xy
screen size 24 by 80
{
f1                  [f000          ]
                    [f000          ]
                    [f000          ]
                    [f000          ]
                    [f000          ]
}
end
tables
tab5
attributes
f000 = tab5.f1, wordwrap compress;
end
```

ZEROFILL
richtet Feldinhalte rechtsbündig aus und füllt das Bildschirmfeld links mit Nullen auf. ZEROFILL ist besonders bei numerischen Feldern sinnvoll. Falls die eingegebene Zahl kürzer als das Bildschirmfeld ist, füllt PERFORM links mit Nullen auf.

## INSTRUCTIONS-Abschnitt

Der INSTRUCTIONS-Abschnitt ist frei wählbar. Er ist notwendig, wenn

- Tabellen über mehr als eine gemeinsame Tabellenspalte verbunden werden sollen (COMPOSITES-Join) oder
- das Suchen in verbundenen Tabellen komfortabler gestaltet werden soll (MASTER/DETAIL-Beziehung) oder
- die Art der Feldbegrenzer verändert werden soll (DELIMITERS-Anweisung) oder
- im Format bestimmte Anweisungen, wie z.B. das Durchführen von Berechnungen, vor oder nach einer PERFORM-Operation, wie z.B. dem Neuaufnehmen oder Suchen, durchgeführt werden sollen (Kontrollblock)

```
INSTRUCTIONS
[tabelle1␣MASTER␣OF␣tabelle2 [;]
[COMPOSITES␣[*]{<[tabelle1.]spalte1,{[tabele1.]spalte2},...>
              ␣[*]<[tabelle2.]spalte1,{[tabelle2.]spalte2},...>[;]
[DELIMITERS␣"feldbegrenzer1 feldbegrenzer2"[;]                    ...
[kontrollblock [;]
[END]
```

MASTER␣OF
: verbindet einen Satz einer Tabelle (der MASTER-Tabelle) beim Suchen mit einem oder mehreren Sätzen einer anderen Tabelle (der DETAIL-Tabelle).

  Wenn Sie in einem Format zwei Tabellen lediglich über einen Join, nicht aber über eine MASTER/DETAIL-Beziehung verbunden haben, erhalten Sie beim Suchen in PERFORM zum ersten Satz einer Tabelle nur den ersten Satz der verbundenen Tabelle. Die anderen Sätze der verbundenen Tabelle erhalten Sie nur durch ein erneutes Suchen in der verbundenen Tabelle.

Wenn Sie hingegen eine MASTER/DETAIL-Beziehung definiert haben, können Sie in PERFORM die Funktionen *Master* und *Detail* anwenden. *Detail* führt die Suche in der Detailtabelle dann automatisch aus.

PERFORM gibt eine Fehlermeldung aus, wenn Sie die Befehle *Master* oder *Detail* mit PERFORM benützen, ohne eine MASTER/DETAIL-Beziehung definiert zu haben.

tabelle1, tabelle2

*tabelle1* und *tabelle2* müssen durch einen Join miteinander verbunden sein. *tabelle1* wird als Master-Tabelle, *tabelle2* wird als Detail-Tabelle definiert. *tabelle1* und *tabelle2* dürfen keine temporären Tabellen sein.

Ist für die Tabelle ein Alias-Name definiert (TABLES-Abschnitt), so müssen Sie hier den Alias-Namen verwenden.

Sie können eine MASTER/DETAIL-Beziehung in beide Richtungen definieren (siehe Beispiel 2).

*Beispiel 1* zu MASTER/DETAIL-Beziehung
(aus dem Formatprogramm *muster*, Anhang A.1):

```
kunde master of auftrag;
auftrag master of posten;
```

Die hier definierte MASTER/DETAIL-Beziehungen ist nützlich, da jeder Kunde mehrere Aufträge haben und jeder Auftrag aus mehreren Posten bestehen kann.

*Beispiel 2* zu MASTER/DETAIL-Beziehung in beide Richtungen

Für eine Tabelle *personal*, die alle Beschäftigten auflistet und eine Tabelle *projekt*, die alle Projekte auflistet, wird eine MASTER/DETAIL-Beziehung in beide Richtungen definiert:

```
personal master of projekt
projekt master of personal
```

Eine sinnvolle Anwendung dieser MASTER/DETAIL-Beziehung könnte so lauten:
Sie wollen für einen Beschäftigten, dessen Projekt Sie nicht kennen, alle Mitarbeiter dieses Projekts suchen.
1. Schritt: Sie aktivieren die Tabelle *personal* und geben dort den Namen des Beschäftigten ein.
2. Schritt: Über *Detail* wechseln Sie dann in die Tabelle *projekt.* und erhalten automatisch Information über dessen Projekt.
3. Schritt: Sie geben widerum *Detail* ein. PERFORM liefert automatisch alle verbundenen Sätze aus *personal*, d. h. alle Beschäftigten dieses Projekts.

COMPOSITES
zeigt an, daß die in spitzen Klammern < > eingeschlossenen Spalten zusammengehörige, miteinander verbundene Spalten sind.

Sie definieren einen COMPOSITES-Join zwischen zwei Tabellen, wenn Sie Werte von *mehr* als einer Tabellenspalte angeben müssen, um einen Satz eindeutig identifizieren zu können.

<[tabelle1.]spalte1, [tabelle1.]spalte2,...>
<[tabelle2.]spalte1, [tabelle2.]spalte2,...>
sind die Spaltennamen der am COMPOSITES-Join beteiligten Spalten.

Ist für die Tabelle ein Alias-Name definiert (TABLES-Abschnitt), so müssen Sie hier den Alias-Namen verwenden.

Die Spaltennamen müssen in spitzen Klammern < > eingeschlossen sein.

Jede Spalte, die Sie in einem COMPOSITES-Join angeben, müssen Sie auch im ATTRIBUTES-Abschnitt über einen Join verbunden haben (siehe ATTRIBUTES-Abschnitt).

Zwischen den verbundenen Tabellen darf keine zusätzliche Verbindung existieren, die nicht im COMPOSITES-Join erfaßt ist.

Die Spalten, die an einem COMPOSITES-Join beteiligt sind, müssen den gleichen Datentyp haben.

Wenn die am COMPOSITES-Join beteiligten Spalten einzeln und gemeinsam indiziert sind, erhöht sich die Ablaufgeschwindigkeit einer Datenbankabfrage.

*

macht Spalten einer Tabelle des COMPOSITES-Joins zu dominanten Spalten. Die anderen am COMPOSITE-Join beteiligten Spalten können dann beim Neuaufnehmen oder Korrigieren nur Werte aufnehmen, die in den dominanten Spalten schon vorhanden sind (überprüfender COMPOSITES-Join).

In der Tabelle, die die dominante Spalte enthält, können Sie einen Satz nur löschen, wenn Sie verbundene Sätze aus anderen Tabellen zuvor gelöscht haben.

In einem COMPOSITES-Join dürfen nur Spalten *einer* Tabelle zu dominanten Spalten gemacht werden.

*Beispiel* zu COMPOSITES
(aus dem Formatprogramm *muster*, Anhang A.1):

```
composites <posten.artikel_nr, posten.herstellercode>
           * <artikel.artikel_nr, artikel.herstellercode>
```

Die Spalten *artikel_nr* und *herstellercode* der Tabellen *artikel* und *posten* bilden einen COMPOSITES-Join. Es handelt sich hierbei um einen überprüfenden COMPOSITES-Join. D.h., wenn die Tabelle *posten* aktiv ist, kann der Benutzer in die Spalten *artikel_nr* und *herstellercode* nur solche Werte eintragen, die bereits beide gemeinsam in einem Satz der Tabelle *artikel* vorhanden sind. Durch den überprüfenden COMPOSITES-Join wird verhindert, daß Artikelnummern und Herstellercodes eingetragen werden können, die zwar als Einzelwerte, nicht aber als gekoppelte Werte gemeinsam in einem Satz der Tabelle *artikel* auftreten.

Im Anhang A.1 finden Sie eine Auflistung der Werte der einzelnen Tabellen des Formatprogramms *muster*. Die Tabelle *artikel* enthält 3 Sätze, in denen die Spalte *artikel_nr* den Wert 1 hat und 4 Sätze, in denen die Spalte *herstellercode* den Wert HRO hat. Man braucht sowohl die Artikelnummer (= 1) als auch den Herstellercode (= HRO), um einen Satz der Tabelle eindeutig zu identifizieren (= von HRO hergestellte Ski-Handschuhe).

DELIMITERS
verändert die Art der Feldbegrenzer.

"feldbegrenzer1 feldbegrenzer2"
bezeichnet den linken bzw. den rechten Feldbegrenzer.

Sie können jedes abdruckbare Zeichen, einschließlich des Leerzeichens, angeben.

Sie dürfen nur ein einzelnes Zeichen pro Feldbegrenzer verwenden und zwischen den beiden Feldbegrenzern kein Leerzeichen ␣ angeben.

Auch wenn Sie die Art des Feldbegrenzers über DELIMITERS verändern, müssen Sie im SCREEN-Abschnitt weiterhin eckige Klammern [] als Feldbegrenzer angeben.

Sie können im SCREEN-Abschnitt statt der eckigen Klammern [] ein beliebiges Zeichen verwenden, sofern Sie damit gleichzeitig sowohl den Schlußbegrenzer von einem Feld als auch den Anfangsbegrenzer von einem anderen Feld kennzeichnen.

*Beispiel* zu DELIMITERS:

```
Name [f000          |f001      ]
```

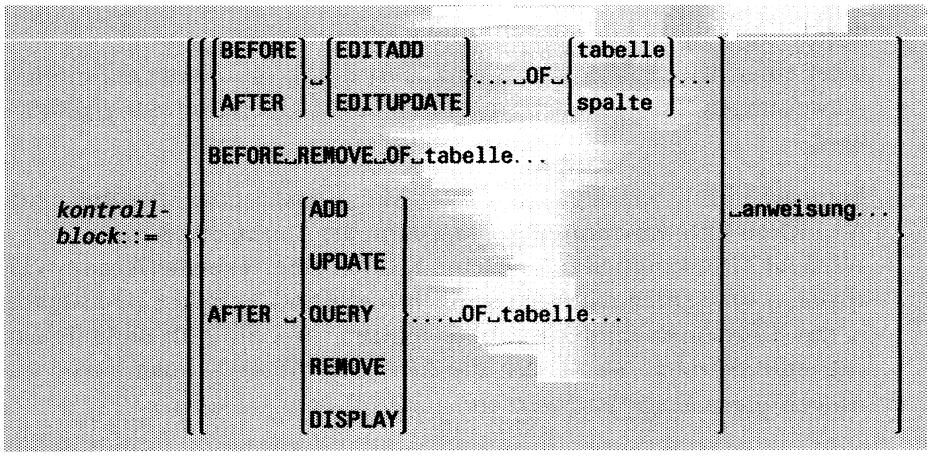
*feldbezeichner1* (f000) kennzeichnet das erste und *feldbezeichner2* (f001) das zweite Bildschirmfeld.

Die entsprechende Anweisung bei Delimiters lautet:

DELIMITERS "||";

### Kontrollblöcke

Kontrollblöcke ermöglichen es, vor oder nach einer PERFORM-Operation bestimmte Aktionen auszuführen, etwa wie "Feldern Werte zuweisen", "Schreibmarke in ein Feld bewegen" usw.




BEFORE

leitet einen Kontrollblock ein und führt die in *anweisung* definierten Anweisungen aus, bevor die PERFORM-Operation ausgeführt wird.

Bei BEFORE beachtet PERFORM zuerst die *anweisung*, die Sie im INSTRUCTIONS-Abschnitt definiert haben und führt dann das im ATTRIBUTES-Abschnitt definierte Attribut aus.
Wird BEFORE lediglich für einzelne Spalten einer Tabelle vereinbart (Angabe OF spalte...), werden die Anweisungen ausgeführt, sobald die Schreibmarke in das entsprechende Bildschirmfeld bewegt wird.

Wenn mehrere Operationen durchgeführt werden sollen, müssen diese durch Leerzeichen voneinander getrennt werden.

Wenn Sie eine Tabellenspalte des Datentyps CHAR in mehrere Bildschirmfelder aufgeteilt haben, führt PERFORM die angegebenen Anweisungen bei jedem einzelnen Bildschirmfeld durch.

- Wenn die Anweisungen nur beim ersten Bildschirmfeld wirksam sein sollen, müssen Sie anstelle des BEFORE-Kontrollblocks einen mit AFTER beginnenden Kontrollblock für das vorhergehende Bildschirmfeld definieren.
- Wenn die Anweisungen nur beim letzten Bildschirmfeld wirksam sein sollen, müssen Sie einen mit BEFORE beginnenden Kontrollblock für das nachfolgende Bildschirmfeld definieren.

AFTER

leitet einen Kontrollblock ein und führt die in *anweisung* definierten Anweisungen aus, nachdem die PERFORM-Operation durchgeführt wird.

Bei AFTER beachtet PERFORM zuerst das Attribut im ATTRIBUTES-Abschnitt und führt dann die Anweisung aus.
Wird AFTER lediglich für einzelne Spalten einer Tabelle (Angabe OF spalte...), so werden die Anweisungen ausgeführt, sobald eine Eingabe in das Bildschirmfeld mit [↵] abgeschlossen wird. Sind für diese Felder Attribute definiert, so werden die Attribute wirksam, noch bovor die Anweisungen ausgeführt werden.

Wenn mehrere Operationen durchgeführt werden sollen, müssen diese durch Leerzeichen voneinander getrennt werden.

AFTER EDITADD OF *tabelle* und AFTER EDITUPDATE OF *tabelle* führt die Anweisungen aus, bevor der Satz in die Datenbank übernommen wird. Dies bietet die Möglichkeit, mit IF Prüfungen vorzunehmen, bevor die Datenbank verändert wird.

AFTER EDITADD wird nur ausgeführt, wenn in das Feld eine Eingabe gemacht wird.

*Beispiel* zu AFTER
(aus dem Formatprogramm *muster*, Anhang A.1):

```
after add update query of posten
   if (total of p19) <= 100 then
      let d1 = 7.50
      else
      let d1 = (total of p19) * .04
   let d2 = (total of p19) + d1
```

Sobald der Benutzer das Neuaufnehmen, Korrigieren oder Suchen in der Tabelle *posten* abgeschlossen und auf [START] gedrückt hat, berechnet PERFORM die Werte für die Bildschirmfelder mit den Feldbezeichnern *d1* (Liefergebühr) und *d2* (Auftragsgesamtsumme) und zeigt diese Werte auf dem Bildschirm an.

EDITADD

bewirkt, daß bei Formatablauf die in *anweisung* definierten Anweisungen vor bzw. nach dem Neuaufnehmen durchgeführt werden.

Bei EDITADD werden diese Anweisungen durchgeführt, *bevor* ein Satz in die Tabelle geschrieben wird.

*Beispiel* zu EDITADD
(aus dem Formatprogramm *muster*, Anhang A.1):

```
after editadd editupdate of menge
   let p19 = p18 * ar15
   nextfield = au11
```

Der Benutzer trägt über die PERFORM-Funktion *Neuaufnehmen* oder *Korrigieren* in das Bildschirmfeld *Menge* einen Wert ein. PERFORM berechnet dann automatisch den Wert des Feldes *Gesamtpreis* (siehe LET im anschließenden Abschnitt *Anweisungen für Kontrollblöcke*) und bewegt die Schreibmarke in das Bildschirmfeld mit dem Feldbezeichner *au11* (siehe NEXTFIELD im anschließenden Abschnitt *Anweisungen für Kontrollblöcke*).

EDITUPDATE

bewirkt, daß bei Formatablauf die in *anweisung* definierten Anweisungen vor bzw. nach dem Korrigieren durchgeführt werden.

Bei EDITUPDATE werden diese Anweisungen durchgeführt, *bevor* ein Satz in die Tabelle geschrieben wird.

```
{tabelle}
{spalte }
```

Sie dürfen maximal 16 Tabellenspalten und/oder Tabellen angeben. Wenn Sie mehrere Tabellen oder Tabellenspalten angeben, müssen Sie diese durch Leerzeichen voneinander abtrennen.

Für Spalte geben Sie den Namen der Tabellenspalte an.

Die Tabelle *displaytable* können Sie nur dann angeben, wenn Sie DISPLAYONLY-Felder durch die Angabe ALLOWING INPUT so definiert haben, daß Eingaben durch den Benutzer erlaubt sind (siehe ATTRIBUTES-Abschnitt).

*anweisung*
finden Sie im anschließenden Abschnitt *Anweisungen für Kontrollblöcke* beschrieben.

ADD
entspricht dem Neuaufnehmen in PERFORM. Die in *anweisung* definierten Anweisungen werden durchgeführt, *nachdem* der Satz in die Tabelle aufgenommen wurde.

UPDATE
entspricht dem Korrigieren in PERFORM. Die in *anweisung* definierten Anweisungen werden durchgeführt, *nachdem* der Satz in die Tabelle zurückgeschrieben wurde.

QUERY
entspricht dem Suchen in PERFORM.

REMOVE
entspricht dem Löschen in PERFORM.

DISPLAY
wird immer durchgeführt, wenn die Daten ausgegeben werden.

*Beispiel* zu DISPLAY
(aus dem Formatprogramm *muster*, Anhang A.1):

```
after display of auftrag
   let d1 = 0
   let d2 = 0
```

Sobald die Werte der Tabelle *auftrag* auf dem Bildschirm angezeigt werden, setzt PERFORM die Werte der Bildschirmfelder *d1* (Liefergebühr) und *d2* (Auftragsgesamtsumme) auf 0 (siehe LET im anschließenden Abschnitt *Anweisungen für Kontrollblöcke*).

tabelle

Sie dürfen maximal 16 Tabellen, einschließlich der Tabelle *displaytable* angeben. Wenn Sie mehrere Tabellen angeben, müssen Sie diese durch Leerzeichen voneinander abtrennen.

Die Tabelle *displaytable* können Sie nur dann angeben, wenn Sie DISPLAYONLY-Felder durch die Angabe ALLOWING INPUT so definiert haben, daß Eingaben durch den Benutzer erlaubt sind (siehe ATTRIBUTES-Abschnitt).

*anweisung*

finden Sie im anschließenden Abschnitt *Anweisungen für Kontrollblöcke* beschrieben.

Die folgende Tabelle zeigt nochmals an, wann Anweisungen ausgeführt werden.

| | BEFORE spalte | AFTER spalte | BEFORE tabelle | AFTER tabelle |
|---|---|---|---|---|
| EDITADD | beim Neuaufnehmen, wenn die Schreibmarke in das Feld eintritt | beim Neuaufnehmen, wenn die Schreibmarke das Feld verläßt | beim Neuaufnehmen bevor ein Format bearbeitet wird | nach START, aber bevor der Satz in die Datenbank neu aufgenommen wird |
| EDITUPDATE | beim Korrigieren wenn die Schreibmarke in das Feld eintritt | beim Korrigieren, wenn die Schreibmarke das Feld verläßt | beim Korrigieren bevor ein Format bearbeitet wird | nach START, aber bevor der Satz in der Datenbank korrigiert wird |
| ADD | nicht möglich | nicht möglich | nicht möglich | nachdem der Satz in die Datenbank neu aufgenommen wurde |
| UPDATE | nicht möglich | nicht möglich | nicht möglich | nachdem der Satz in der Datenbank korrigiert wurde |
| REMOVE | nicht möglich | nicht möglich | bevor der Satz aus der Datenbank gelöscht wird | nachdem der Satz aus der Datenbank gelöscht wurde |
| DISPLAY | nicht möglich | nicht möglich | nicht möglich | jedesmal, wenn die Daten ausgegeben werden |
| QUERY | nicht möglich | nicht möglich | nicht möglich | nach Ausführen eines Suchkommandos |

**Hinweis**

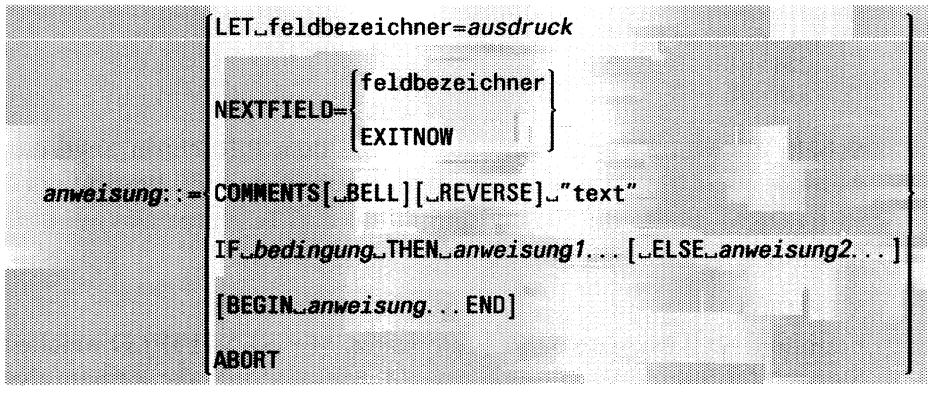
- Bei ADD, UPDATE, QUERY, REMOVE und DISPLAY werden Anweisungen durchgeführt, nachdem der Satz von PERFORM in die Tabelle geschrieben wurde.
- Bei EDITADD und EDITUPDATE werden Anweisungen durchgeführt, bevor ein Satz von PERFORM in die Tabelle geschrieben wird.

### Anweisungen für Kontrollblöcke

Folgende *anweisungen* können Sie innerhalb eines Kontrollblocks durchführen lassen:

- Feldern Werte zuweisen
- die Schreibmarke in ein Feld bewegen
- eine Meldung anzeigen
- eine der eben genannten Anweisungen in Abhängigkeit von bestimmten Bedingungen durchführen und
- PERFORM-Operationen beenden und in das PERFORM-Menü zurückkehren

Alle Anweisungen müssen innerhalb eines BEFORE- oder AFTER-Kontrollblocks definiert werden.




LET
: weist einem Bildschirmfeld einen Wert zu. Der Wert kann eine Konstante sein oder PERFORM kann ihn aus der aktuellen Liste berechnen.

feldbezeichner
: ist der Feldbezeichner des Bildschirmfeldes, dem Sie einen Wert zuweisen wollen.

Das Bildschirmfeld darf keiner Tabellenspalte des Datentyps SERIAL entsprechen.

Das Bildschirmfeld muß

- zur Liste der Spalten oder Tabellen gehören, für die LET ausgeführt wird,
- ein Feld sein, über das alle in der Liste genannten Tabellen verbunden sind oder
- ein DISPLAYONLY-Feld sein

Sie können nur Spalten der aktiven Tabelle oder DISPLAYONLY-Feldern Werte zuweisen.

*ausdruck*
finden Sie im anschließenden Abschnitt *Ausdrücke für Anweisungen* beschrieben.

*Beispiel* zu LET
(aus dem Formatprogramm *muster*, Anhang A.1):

```
after display of auftrag
   let d1 = 0
   let d2 = 0
```

Sobald die Werte der Tabelle *auftrag* auf dem Bildschirm angezeigt werden, wird die Anweisung LET durchgeführt. LET bewirkt, daß PERFORM die Werte der Bildschirmfelder *d1* (Liefergebühr) und *d2* (Auftragsgesamtsumme) auf 0 setzt.

NEXTFIELD
bestimmt die Bewegung der Schreibmarke. Mit NEXTFIELD weichen Sie von der Standardpositionierung ab, wie sie durch die Reihenfolge der Feldbezeichner im ATTRIBUTES-Abschnitt vorgegeben ist.
NEXTFIELD ist nur bei EDITUPDATE und EDITADD sinnvoll.

feldbezeichner
bezeichnet das Bildschirmfeld, in das die Schreibmarke in PERFORM automatisch positioniert werden soll.
Das Bildschirmfeld muß zur aktuellen Tabelle gehören.

EXITNOW
geht zum PERFORM-Menü zurück, ohne daß Sie die Taste START drücken müssen. Vor Rückkehr in das PERFORM-Menü werden ggf. folgende PERFORM-Funktionen ausgeführt: *Neuaufnehmen*, *Korrigieren*, *Loeschen*. Vergleiche dazu: Die Anweisung ABORT bricht den Formatablauf ab, ohne vorher diese PERFORM-Funktionen durchzuführen.

*Beispiel* zu NEXTFIELD
(aus dem Formatprogramm *muster*, Anhang A.1):

```
before editadd editupdate of auftrag
   nextfield = au20
```

Noch bevor der Benutzer in der aktiven Tabelle *auftrag* neuaufnehmen oder korrigieren kann, bewegt PERFORM die Schreibmarke in das Bildschirmfeld mit dem Feldbezeichner *au20* (*fremd_nr*). Ohne die Anweisung NEXTFIELD würde die Schreibmarke zuerst in die erste Spalte der Tabelle *auftrag*, das Bildschirmfeld *au11* gehen, da dieser Feldbezeichner im ATTRIBUTES-Abschnitt des Formatprogramms *muster* vor dem Feldbezeichner *au20* genannt wird.

COMMENTS
gibt am unteren Bildschirmrand, in der Statuszeile, eine Meldung aus. (COMMENTS im ATTRIBUTES-Abschnitt schreibt eine Meldung in die Kommentarzeile.) COMMENTS ist nur sinnvoll bei EDITUPDATE und EDITADD.

BELL
macht mit einem akustischen Signal auf die Meldung aufmerksam.

REVERSE
bildet die Meldung invers ab.

"text"
ist der Text Ihrer Meldung. Die Meldung darf nicht länger als eine Bildschirmzeile sein.

IF

leitet eine Bedingung ein. Abhängig von dieser Bedingung werden Anweisungen ausgeführt:

- ist die Bedingung erfüllt, werden die Anweisungen des THEN-Zweigs ausgeführt
- ist die Bedingung nicht erfüllt, werden die Anweisungen des ELSE-Zweigs ausgeführt

*bedingung*

finden Sie im INSTRUCTIONS-Abschnitt unter *Bedingungen für Anweisungen* beschrieben.

THEN

leitet nach Erfüllen einer Bedingung *anweisung1* ein.

*anweisung1*

wird ausgeführt, wenn die Bedingung erfüllt ist. Wenn Sie nicht nur eine, sondern mehrere Anweisungen ausführen wollen, **müssen** Sie diese Anweisungen mit BEGIN einleiten und mit END beenden. *BEGIN␣anweisung... END* klammert mehrere Anweisungen zu einer Anweisung zusammen.

ELSE

leitet eine Anweisung ein.

*anweisung2*

wird ausgeführt, wenn die Bedingung nicht erfüllt ist. Wenn Sie nicht nur eine, sondern mehrere Anweisungen ausführen wollen, **müssen** Sie diese Anweisungen mit BEGIN einleiten und mit END beenden. *BEGIN␣anweisung... END* klammert mehrere Anweisungen zu einer Anweisung zusammen.

*Beispiel*

(aus dem Formatprogramm *muster*, Anhang A.1):

```
after add update query of posten

   if (total of p19) == 100 then
      let d1 = 7.50
      else
      let d1 = (total of p19) * .04

   let d2 = (total of p19) + d1
```

Sobald der Benutzer das Neuaufnehmen, Korrigieren oder Suchen der Tabelle *posten* mit START abgeschlossen hat, berechnet PERFORM die Werte für die Bildschirmfelder mit den Feldbezeichnern *d1* und *d2* und zeigt diese Werte auf dem Bildschirm an.

Die Werte für die Felder *Liefergebühr* und *Auftragsgesamtsumme* werden dabei auf folgende Weise berechnet:

Der Wert für das Feld *Liefergebühr* wird in Abhängigkeit von einer Bedingung berechnet:

- wenn die Summe des Feldes *Gesamtpreis* aller Posten eines Auftrags (alle Posten mit gleicher Auftragsnummer) kleiner oder gleich 100 ist, erhält das Bildschirmfeld mit dem Feldbezeichner *d1* (Liefergebühr) den Wert '7.50'
- wenn diese Bedingung nicht erfüllt ist, wird *Liefergebühr* wie folgt berechnet:
  - die Werte des Feldes *Gesamtpreis* aller Posten eines Auftrags (mit gleicher Auftragsnummer) werden addiert
  - der Wert des Feldes *Liefergebühr* entspricht dann 4% der so gebildeten Summe

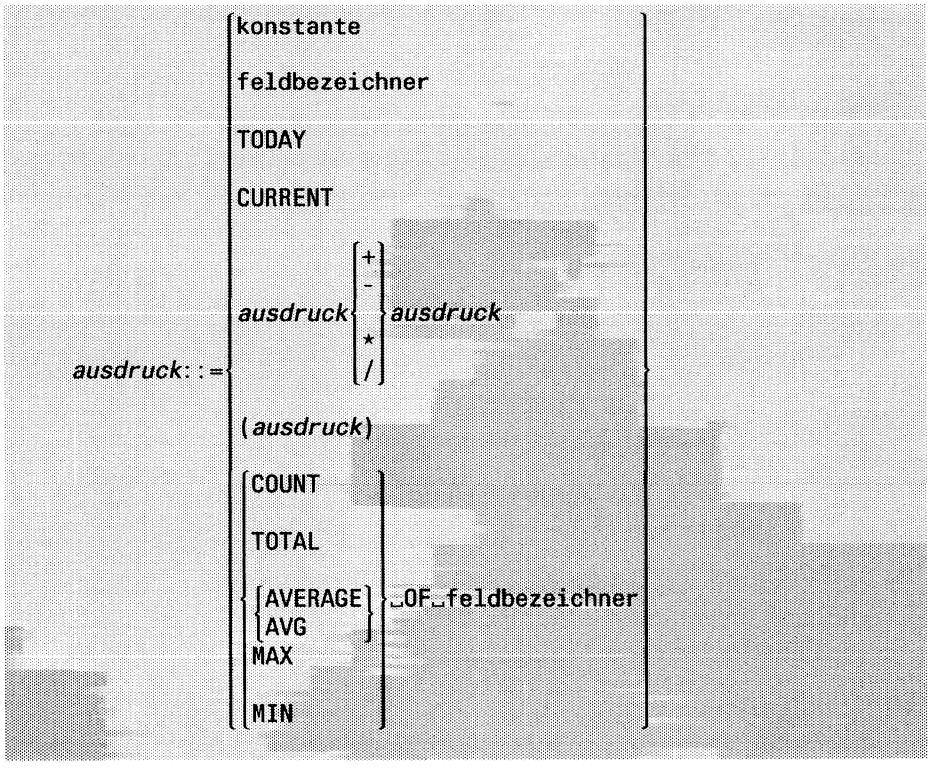
Der Wert für das Bildschirmfeld *Auftragsgesamtsumme* errechnet sich durch die Addition der Summe des Feldes *Gesamtpreis* mit dem zuvor errechneten Wert von *Liefergebühr.*

BEGIN *anweisung...* END
: geben Sie an, wenn Sie statt einer Anweisung mehrere Anweisungen durchführen wollen. BEGIN... END klammert mehrere Anweisungen zu einer Anweisung zusammen.

ABORT
: bricht den Formatablauf ab und kehrt in das PERFORM-Menü zurück, ohne die PERFORM-Funktionen *Neuaufnehmen, Korrigieren* oder *Löschen* durchzuführen. ABORT entspricht der Taste [DEL] und darf nur bei EDITUPDATE, EDITADD und REMOVE benützt werden.
  Vergleiche dazu: Die Anweisung NEXTFIELD EXITNOW führt PERFORM-Funktionen noch vor Rückkehr in das PERFORM-Menü aus.

### Ausdrücke für Anweisungen

Ausdrücke benötigen Sie, wenn Sie Bedingungen formulieren.




konstante
: Eine alphanumerische Konstante oder ein Datum muß in Anführungszeichen eingeschlossen sein. Kommazahlen müssen den Dezimalpunkt enthalten, wenn Kommastellen existieren.

Die Form der Datumangabe hängt von der Umgebungsvariablen DBDATE ab (siehe Kapitel 7.2).

feldbezeichner
: ist ein Feldbezeichner, den Sie im SCREEN-Abschnitt angegeben haben.

TODAY
ist das aktuelle Tagesdatum bezogen auf den Eingabetag.

CURRENT
ist das aktuelle Tagesdatum und die Uhrzeit bezogen auf den Eingabetag.

{+, -, *, / }
kennzeichnen die Rechenarten Addieren, Subtrahieren, Multiplizieren und Dividieren.

*(ausdruck)*
Die Klammern brauchen Sie, um Teile von Ausdrücken zu einer Einheit zusammenzufassen. Damit können Sie abweichend von den üblichen Vorrangregeln die Reihenfolge bestimmen, in der Ausdrücke ausgewertet werden sollen.

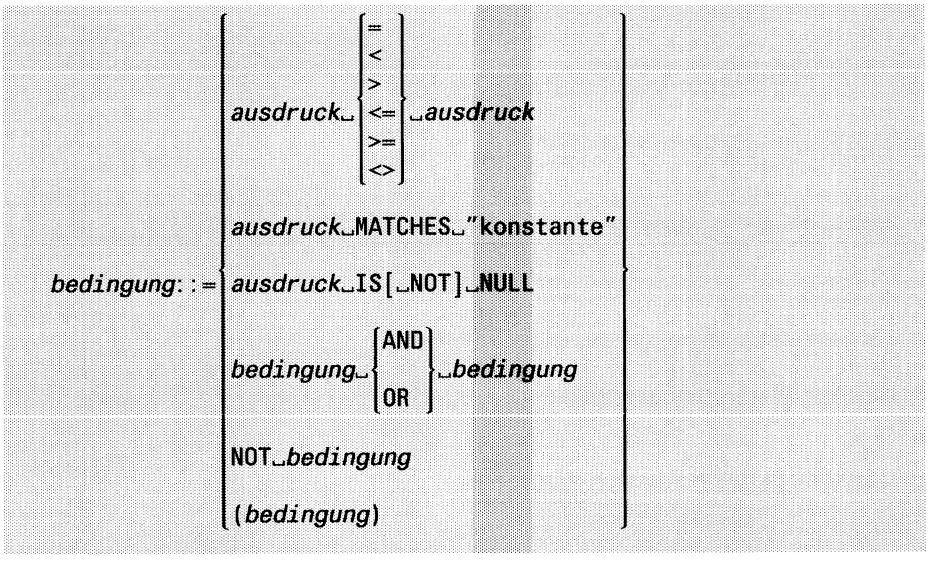
COUNT, TOTAL, AVG, AVERAGE, MAX, MIN
sind Mengenfunktionen.

| **Mengenfunktion** | **Bedeutung** |
|---|---|
| COUNT | gibt die Anzahl von Sätzen der aktuellen Liste aus. |
| TOTAL | summiert die Werte des angegebenen Feldes für alle Sätze der aktuellen Liste. Das Feld muß numerische Werte enthalten. |
| [AVG \| AVERAGE] | errechnet aus den Werten des angegebenen Feldes den Mittelwert. Das Feld muß numerische Werte enthalten. |
| MAX | liefert den höchsten Wert. |
| MIN | liefert den niedrigsten Wert. |

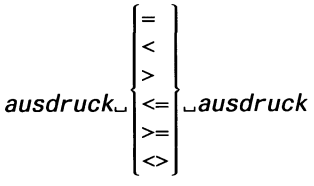
feldbezeichner
ist ein Feldbezeichner, den Sie im SCREEN-Abschnitt angegeben haben. Das zugehörige Bildschirmfeld darf kein DISPLAYONLY-Feld sein. Die Rechenoperationen beziehen sich in PERFORM auf die aktuelle Liste und das Bildschirmfeld, dem Sie den Feldbezeichner zugewiesen haben.

### Bedingungen für Anweisungen

*bedingung* prüft Zustände ab, bevor eine Anweisung bei Erfüllen dieser Bedingung durchgeführt wird.




*ausdruck*
ist im Abschnitt *Ausdrücke für Anweisungen* erklärt.




Vergleicht das Ergebnis eines Ausdrucks mit dem eines anderen Ausdrucks.

| Vergleichsoperator | Bedeutung |
|---|---|
| = | gleich |
| < | kleiner |
| > | größer |
| <= | kleiner oder gleich |
| >= | größer oder gleich |
| <> | ungleich |

*Die Bedingung ist erfüllt*, wenn der entsprechende Vergleich stimmt.

*Hinweis:*

Wenn Sie Ausdrücke vom Typ CHAR vergleichen, so gilt die ASCII-Reihenfolge der einzelnen Zeichen: Ein Zeichen ist kleiner als ein anderes Zeichen, wenn es in der ASCII-Tabelle vor diesem Zeichen steht. Ein Zeichen ist größer als ein anderes Zeichen, wenn es in der ASCII-Tabelle nach diesem Zeichen steht. (Eine ASCII-Tabelle finden Sie im Anhang A.3)

Wenn Sie Ausdrücke vom Typ DATE vergleichen, so gilt:
Ein Datum ist kleiner als ein anderes Datum, wenn es älter ist. Ein Datum ist größer als ein anderes Datum, wenn es jünger ist.

*Beispiel* zu *ausdruck* <= *ausdruck*
(aus dem Formatprogramm *muster*, Anhang A.1):

```
after add update query of posten

   if (total of p19) <= 100 then
      let d1 = 7.50
      else
      let d1 = (total of p19) * .04

   let d2 = (total of p19) + d1
```

Sobald der Benutzer das Neuaufnehmen, Korrigieren oder Suchen der Tabelle *posten* mit START abgeschlossen hat, berechnet PERFORM die Werte für die Bildschirmfelder mit den Feldbezeichnern *d1* und *d2* und zeigt diese Werte auf dem Bildschirm an.

Die Werte für die Felder *Liefergebühr* und *Auftragsgesamtsumme* werden dabei auf folgende Weise berechnet:

Der Wert für das Feld *Liefergebühr* wird in Abhängigkeit von einer Bedingung berechnet:

- wenn die Summe des Feldes *Gesamtpreis* aller Posten eines Auftrags (alle Posten mit gleicher Auftragsnummer) kleiner oder gleich 100 ist, erhält das Bildschirmfeld mit dem Feldbezeichner *d1* (Liefergebühr) den Wert '7.50'
- wenn diese Bedingung nicht erfüllt ist, wird *Liefergebühr* wie folgt berechnet:
  - die Werte des Feldes *Gesamtpreis* aller Posten eines Auftrags (mit gleicher Auftragsnummer) werden addiert
  - der Wert des Feldes *Liefergebühr* entspricht dann 4% der so gebildeten Summe

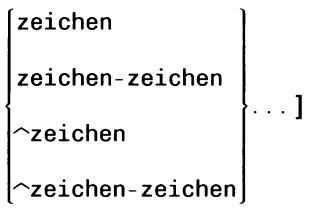
Der Wert für das Bildschirmfeld *Auftragsgesamtsumme* errechnet sich durch die Addition der Inhalts im Feld 'Gesamtpreis' mit dem zuvor errechneten Wert von *Liefergebühr.*

*ausdruck*␣MATCHES␣"konstante"
brauchen Sie für Vergleiche mit einer Konstanten, deren Wert Sie nicht eindeutig angeben können.

Sie dürfen diese Bedingung nur bei Feldern vom Datentyp CHAR verwenden.

In der nachfolgenden Tabelle ist dargestellt, wie eine Konstante aufgebaut sein kann.

| **Konstante enthält** | **Gesucht wird jeder Feldinhalt, der an derselben Stelle** |
|---|---|
| zeichen | genau das Zeichen enthält |
| ? | ein beliebiges Zeichen enthält |
| * | eine beliebige Zeichenfolge oder nichts enthält. |




ein Zeichen enthält, das zu den angegebenen Zeichen oder Zeichenbereichen gehört. ^ bezieht sich auf alle nachfolgenden Zeichen und gibt ihnen folgende Bedeutung: ein Zeichen enthält, das nicht zu den angegebenen Zeichen oder Zeichenbereichen gehört. Die eckigen Klammern müssen Sie eingeben.

* * enthält

\? ? enthält

*ausdruck*␣IS[␣NOT]␣NULL

Prüft, ob der angegebene Ausdruck NULL-Werte bzw. keine NULL-Werte enthält. Die Bedingung ist erfüllt, wenn der angegebene Ausdruck NULL-Werte enthält bzw. wenn er keine NULL-Werte enthält (Angabe NOT).

*bedingung*␣{AND / OR}␣*bedingung*

NOT␣*bedingung*

Verknüpft eine oder mehrere Bedingungen mit logischen Operatoren.

AND
: logisches UND: beide mit AND verknüpften Bedingungen müssen erfüllt sein, damit *Bedingung erfüllt* gilt.

OR
: logisches ODER: eine der beiden mit OR verknüpften Bedingungen muß erfüllt sein, damit *Bedingung erfüllt* gilt.

NOT
: Negation: die mit NOT verknüpfte Bedingung darf nicht erfüllt sein, damit *Bedingung erfüllt* gilt.

**Hinweis**

Wenn Sie die Operatoren AND, OR und NOT kombinieren, so gelten die üblichen Vorrangregeln für die Auswertung: NOT vor AND vor OR. Wenn Sie die beschriebene Reihenfolge ändern möchten, so müssen Sie entsprechend Klammern setzen: Operatoren innerhalb der Klammern haben Vorrang.

*(bedingung)*

Die Klammern brauchen Sie, wenn Sie die Operatoren AND, OR und NOT nicht in der Reihenfolge NOT vor AND vor OR auswerten wollen. Die Operatoren in Klammern haben Vorrang.

# 4 PERFORM - Ein Format ablaufen lassen



PERFORM startet ein Formatprogramm und erzeugt daraus ein Format, das Sie mit speziellen PERFORM-Funktionen bedienen: Sätze suchen, korrigieren, löschen oder neuaufnehmen.

Das Formatprogramm, das die Eingabe für PERFORM bildet, muß zuvor mit FORMBUILD (siehe Kapitel 3) erstellt und übersetzt worden sein.

Die Betriebssystemdatei, die das ablauffähige Formatprogramm enthält, muß entweder im aktuellen oder über DBPATH vereinbarten Dateiverzeichnis enthalten sein (Umgebungsvariable DBPATH, siehe Kapitel 7.2).

Der Aufruf von PERFORM ist sowohl über INFORMIX-Menüs, als auch auf Betriebssystemebene möglich.

## 4.1 Aufruf über INFORMIX-Menüs

Wenn Sie im Menü INFORMIX-SQL die Funktion *Format* auswählen, erhalten Sie das Menü FORMAT.

```
FORMAT:  Ablauf Modifizieren Generieren Neu Compilieren Loeschen END
```

Ablauf

PERFORM-Aufruf: startet ein mit FORMBUILD übersetztes Formatprogramm. Am Bildschirm erscheint eine Liste aller zugreifbaren Formatprogramme. Die Datei, die das ablauffähige Formatprogrmm (*.frm*) enthält, muß im aktuellen Dateiverzeichnis oder in dem über die

Umgebungsvariable DBPATH vereinbarten Dateiverzeichnis enthalten sein (DBPATH siehe Kapitel 7.2).

Die Funktionen *Modifizieren*, *Generieren*, *Neu* und *Compilieren* beziehen sich auf das Erstellen, Ändern oder Übersetzen von Formatprogrammen und sind im Kapitel 3, FORMBUILD beschrieben.

Loeschen
: löscht ein Formatprogramm. Am Bildschirm erscheint eine Liste aller zugreifbaren Formatprogramme. Die Betriebssystemdateien (*.per* und *.frm*) müssen im aktuellen Dateiverzeichnis oder in dem über die Umgebungsvariable DBPATH vereinbarten Dateiverzeichnis enthalten sein, damit das Löschen möglich ist (DBPATH siehe Kapitel 7.2). Nachdem Sie das gewünschte Formatprogramm ausgewählt haben, müssen Sie nochmals bestätigen, daß das Formatprogramm gelöscht werden soll.

END
: beendet das Menü FORMAT und kehrt zum INFORMIX-Hauptmenü zurück.

Wenn Sie im Menü FORMAT die Funktion *Ablauf* wählen, erscheint folgendes Menü:

```
ABLAUF FORMAT >>
Format auswaehlen oder Namen eingeben. Weiter mit ↵
```

Nachdem Sie das gewünschte Formatprogramm ausgewählt haben, erhalten Sie das PERFORM-Menü:

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                              (Tabelle  1:      )
```

Eine Beschreibung der einzelnen Funktionen des PERFORM-Menüs finden Sie im Abschnitt 4.5.

## 4.2 Aufruf auf Betriebssystemebene

Es gibt zwei Möglichkeiten, ein Formatprogramm auf Betriebssystemebene zu starten:

- entweder über den PERFORM-Aufruf *sperform* oder
- über einen *isql*-Aufruf mit anschließender Auswahl des PERFORM-Menüs

Die zweite Möglichkeit, das Starten eines Formatprogramms über einen *isql*-Aufruf, finden Sie in Kapitel 7.3.1 beschrieben.

```
sperform datei...
```

datei

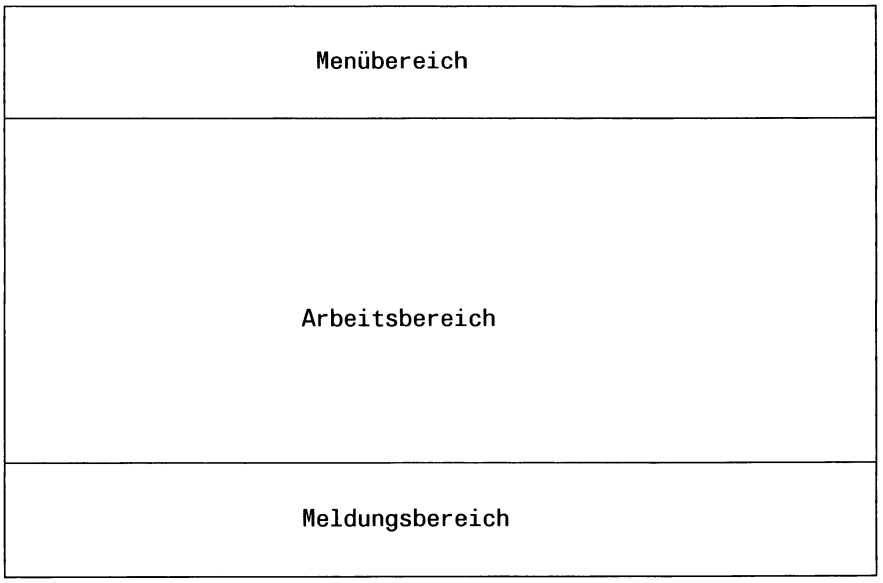
ist der Name des übersetzten Formatprogramms. Die Datei, die das ablauffähige Formatprogramm enthält, hat das Suffix *.frm*. Das Suffix *.frm* geben Sie nicht an.

PERFORM sucht die Datei im aktuellen Dateiverzeichnis. Wenn es sie dort nicht findet, wertet es die Umgebungsvariable DBPATH aus (siehe Kapitel 7.2). DBPATH muß dann anzeigen, wo sich die Datei befindet.

Wenn Sie mehrere Dateien angeben, müssen Sie diese durch Leerzeichen voneinander trennen. PERFORM gibt die Formate entsprechend der im Programmaufruf genannten Reihenfolge aus. Wenn PERFORM ein Format nicht anzeigen kann, wird abgebrochen. Eventuell nachfolgende Formate können dann ebenfalls nicht mehr angezeigt werden.

## 4.3 Aufbau des PERFORM-Bildschirms

Der PERFORM-Bildschirm ist in drei Bereiche eingeteilt:




- Die ersten zwei Zeilen (Menübereich) zeigen
  - die PERFORM-Menüfunktionen, wobei eine Funktion markiert dargestellt ist
  - eine Information zu der markierten Menüfunktion und
  - die Nummer und den Namen der aktuellen Datenbanktabelle.
- Der Arbeitsbereich zeigt das ausgewählte Format.
- Der Meldungsbereich besteht aus
  - einer Kommentarzeile und
  - einer Statuszeile.

### Der Menübereich

Die erste Zeile zeigt in einer Reihe Menüfunktionen, die zweite Zeile eine Beschreibung der markierten Funktion, zusammen mit Nummer und Namen der aktuellen Datenbanktabelle.

Die Anzahl der Funktionen, die gleichzeitig in einer Zeile gezeigt werden können, ist bildschirmabhängig. Bei einer Bildschirmbreite von 80 Zeichen sind die PERFORM-Menüfunktionen auf zwei Zeilen verteilt.

Das PERFORM-Menü hat dann folgendes Aussehen:

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                              (Tabelle  1:       )

```

```
PERFORM:... Loeschen Tabelle  Format  Aktuell  Master  Detail  PRINT  END
                                              (Tabelle  1:       )

```

Mit der Pseudofunktion '...'wechseln Sie zwischen diesen Zeilen hin und her.

Beschreibung der übrigen Menüfunktionen siehe Abschnitt 4.5.

### Der Arbeitsbereich

Im Arbeitsbereich werden die einzelnen Bildschirmfelder des Formats angezeigt. Ein Format kann Spalten aus mehreren Tabellen anzeigen, die aber alle zu einer Datenbank gehören müssen. Ein Format kann aber auch Bildschirmfelder enthalten, die keiner Tabellenspalte entsprechen. Es handelt sich dann um DISPLAYONLY-Felder.

Der Benutzer arbeitet nur innerhalb der Bildschirmfelder. Die Bildschirmfelder werden standardmäßig durch eckige Klammern begrenzt. Falls die einzelnen Bildschirmfelder durch andere Zeichen begrenzt sind, wurde hierfür eine spezielle Anweisung im Formatprogramm definiert (siehe Kapitel 3.3, INSTRUCTIONS-Abschnitt, DELIMITERS).

Manche Bildschirmfelder sind nicht von Feldbegrenzern eingeschlossen. Entweder zeigen sie dann Daten aus Tabellen an, die nicht aktiv sind oder es handelt sich um LOOKUP- oder DISPLAYONLY-Felder (siehe Kapitel 3.3, ATTRIBUTES-Abschnitt, LOOKUP-Felder und DISPLAYONLY-Felder). Sie können in diese Felder nichts eintragen bzw. ändern, es sei denn, Sie haben ALLOWING INPUT im Attributes-Abschnitt spezifiziert.

Ein Format kann länger als der auf dem Bildschirm angezeigte Arbeitsbereich sein. Sowohl die Menüfunktion *Format* im PERFORM-Menü als auch die Taste [⇩] zeigen weitere Felder des Formats an.

Wenn man eine Menüfunktion gewählt hat, springt die Schreibmarke sofort zum Anfang des ersten Formatfeldes.

### Der Meldungsbereich

PERFORM zeigt im Meldungsbereich

- Meldungen, die aufgrund des Formatprogramms erscheinen (siehe Kapitel 3.3, ATTRIBUTES- und/oder INSTRUCTIONS-Abschnitt) oder
- Fehlermeldungen, die PERFORM ausgibt

## 4.4 Mit einem Format arbeiten

Dieses Kapitel beschreibt, mit welchen Tasten Sie im Format

- die Schreibmarke positionieren
- einen Feldinhalt löschen
- einen Feldinhalt rekonstruieren

In folgenden Fällen gelten Abweichungen von der allgemeinen Bedienung:

- Bildschirmfelder, für die im Formatprogramm das WORDWRAP-Attribut vergeben wurde, benutzen einen eigenen Editor, den Multiline-Editor. Die Beschreibung finden Sie in Abschnitt 4.4.4.
- Bildschirmfelder, die dem INFORMIX-ONLINE-Datentypen TEXT oder BYTE zugeordnet sind, lassen sich nicht wie die übrigen Bildschirmfelder bearbeiten. Abschnitt 4.4.5 beschreibt die Bedienung dieser Felder.

### 4.4.1 Schreibmarke positionieren

Zur Positionierung der Schreibmarke benutzen Sie folgende Tasten:

| Taste | Wirkung |
|---|---|
| [↵] [↷] [⇥] | führt eine Syntaxprüfung des Feldinhalts durch. Wenn kein Fehler auftritt, geht die Schreibmarke vorwärts zum Anfang des nächsten Bildschirm-Feldes. |
| [↶] [⇤] | führt eine Syntaxprüfung des Feldinhalts durch. Wenn kein Fehler auftritt, geht die Schreibmarke zurück zum Anfang des vorherigen Bildschirmfelds. |
| [→] [CTRL] [L] | geht innerhalb eines Bildschirm-Feldes ein Zeichen vorwärts.<br><br>Am Ende eines Feldes wird eine Syntaxprüfung durchgeführt. Sofern keine Fehler auftreten, springt die Schreibmarke an den Anfang des nächsten Feldes. Wenn Fehler auftreten, ertönt ein Signal. |

| | |
|---|---|
| ← <br> ⌫ <br> CTRL H | geht innerhalb eines Bildschirm-Feldes ein Zeichen rückwärts. <br><br> Am Anfang eines Feldes wird eine Syntaxprüfung durchgeführt. Sofern keine Fehler auftreten, bewegt sich die Schreibmarke in das vorhergehende Feld. Wenn Fehler auftreten, ertönt ein Signal. Die Schreibmarke bleibt am Anfang des Feldes stehen. |
| ↓ <br> CTRL F | geht vorwärts zum ersten Feld der nächsten Zeile. |
| ↑ <br> CTRL B | geht zurück zum ersten Feld der vorherigen Zeile. |
| ↖ | geht zum ersten Bildschirmfeld des aktuellen Bildschirms. |
| CHAR INSERT | fügt auf der Schreibmarkenposition ein Zeichen Zeichen ein. |
| ⇩ | geht zum ersten Bildschirmfeld einer möglichen nachfolgenden Bildschirmseite. |
| ⇧ | geht zum letzten Bildschirmfeld einer vorhergehenden Bildschirmseite |
| F15 <br> CTRL A | wechselt zwischen Overwrite- und Insert-Modus. Zu Beginn des PERFORM-Aufrufs ist ist automatisch Overwrite-Modus eingestellt. |

### 4.4.2 Feldinhalt löschen

Zum Löschen des Feldinhalts benutzen Sie folgende Tasten:

| **Taste** | **Wirkung** |
|---|---|
| WORD DELETE <br> CTRL D | löscht den Feldinhalt ab der Schreibmarkenposition. |
| CHAR DELETE <br> CTRL X | löscht das Zeichen, unter dem die Schreibmarke steht. |
| F17 <br> CTRL C | löscht beim Suchen alle Feldinhalte der Bildschirmseite. |

### 4.4.3 Feldinhalt rekonstruieren

Mit der Taste [WORD INSERT] oder der Tastenkombination [CTRL] [P] können Sie beim Neuaufnehmen und Korrigieren einen Feldinhalt rekonstruieren.

*Korrigieren*:
Die Korrektur wird rückgängig gemacht und ein evtl. vor Korrektur eingeblendeter Wert angezeigt. Die Schreibmarke springt zum nächsten Bildschirmfeld.

*Neuaufnehmen*:
Der zuletzt im Feld ausgegebene Wert wird angezeigt.

### 4.4.4 Multiline-Editor: Bearbeiten eines WORDWRAP-Feldes

Der *Multiline-Editor* dient zur Bearbeitung eines Bildschirmfeldes, dem das Attribut WORDWRAP zugeordnet wurde (siehe Kapitel 3.3, ATTRIBUTES-Abschnitt). Dieses Bildschirmfeld besteht aus mehreren Feldteilen, man spricht auch von einem mehrzeiligen Bildschirmfeld.

**Merkmale**

Es gibt 2 Arbeitsmodi: Overwrite und Insert. Den Wechsel zwischen Overwrite und Insert führen Sie mit der Taste [F15] durch. Standardmäßig ist der Overwrite-Modus eingestellt.

Blinde Eingaben sind nicht möglich. Wenn Sie am Ende des Bildschirmfeldes versuchen, eine Eingabe zu machen, ertönt ein Signal. Textteile, die über das Feldende hinausgeschoben werden, sind verloren.

**Zeilenumbruch**

Automatischer Zeilenumbruch auf Wortebene: Reicht der Platz am Ende einer Zeile nicht mehr aus, um ein ganzes Wort aufzunehmen, so wird das Wort automatisch an den Anfang der nächsten Zeile bewegt. Werden in einer Zeile Zeichen gelöscht, so wird falls möglich, der nachfolgende Text hochgezogen.

Als Wort gilt jede zusammenhängende Zeichenfolge ≠ '␣', d.h. jede Zeichenfolge zwischen 2 Leerzeichen wird als Wort betrachtet.

Der durch den Umbruch entstehende Freiplatz am Zeilenende wird automatisch mit Leerzeichen '␣' aufgefüllt. Wenn Sie verhindern wollen, daß diese Leerzeichen beim Einspeichern mit aufgenommen werden, müssen Sie zusätzlich zu WORDWRAP die Angabe COMPRESS machen (siehe Kapitel 3.3, ATTRIBUTES-Abschnitt).

#### Auswirkungen des Zeilenumbruchs bei Anzeige und Eingabe von Werten

Der Zeilenumbruch findet sowohl bei der Eingabe von Werten (Funktionen *Neuaufnehmen* und *Korrigieren*) als auch bei der Anzeige von Werten (nach *Suchen*) statt.

#### Auswertung des Feldinhalts beim Einspeichern

Wenn Sie in der Länge Ihres Bildschirmfeldes nicht genügend Platz für zusätzliche (vom Editor automatisch erzeugte) Leerzeichen eingeplant haben, dann kann dies zu Problemen führen:

Ein Wert, der in seiner Länge in das Bildschirmfeld passen würde, wird ist nach *Suchen* im Bildschirmfeld umbrochen dargestellt. Aufgrund der beim Umbruch eingefügten Leerzeichen verlängert sich der Wert entsprechend und kann nicht mehr in voller Länge dargestellt werden. Wenn Sie einen solchen Wert abändern (*Korrigieren*), dann wird beim Zurückschreiben des Satzes nur der angezeigte Teil berücksichtigt, der abgeschnittene Rest ist verloren.

Die Auswertung des eingegebenen Feldinhalts geschieht in folgender Reihenfolge:

1. Zunächst wird der gesamte Inhalt des Bildschirmfeldes gelesen
2. Dann wird COMPRESS ausgeführt, falls angegeben (Kürzung des Wertes um die vom Editor automatisch erzeugten Leerzeichen).
3. Die anschließende Behandlung hängt ab von der Länge des Bildschirmfeldes:

   a) Länge des Bildschirmfeldes = Länge der Tabellenspalte:

   | | |
   |---|---|
   | *Neuaufnehmen:* | Der gesamte Feldinhalt wird in die Tabellenspalte eingespeichert. |
   | *Korrigieren:* | Der Inhalt der Tabellenspalte wird durch den Feldinhalt überschrieben. |

b) Länge des Bildschirmfeldes > Länge der Tabellenspalte:

Behandlung wie unter a) mit folgender Einschränkung:
Der Feldinhalt wird ggf. auf die Länge der Tabellenspalte gekürzt.

c) Länge des Bildschirmfeldes < Länge der Tabellenspalte:

*Neuaufnehmen:* wie bei a)

*Korrigieren:* Der Spaltenwert wird genau in Länge des Bildschirmfeldes überschrieben. D. h. ein evtl. vorhandener Spaltenrest bleibt erhalten und wird nicht nach links verschoben, falls der Inhalt des Bildschirmfeldes entsprechend kürzer ist. In diesem Fall wird mit '␣' auf die Länge des Bildschirmfeldes aufgefüllt und der Spaltenrest daran angehängt.

**Tasten zur Bedienung des WORDWRAP-Feldes**

| Taste | Bedeutung |
|---|---|
| ↵ | Die Schreibmarke springt an den Anfang des nächsten Bildschirmfeldes. Für das letzte Bildschirmfeld des Formates gilt: springt an den Anfang des ersten Bildschirmfeldes im Format. |
| ←<br>CTRL H | bewegt die Schreibmarke um eine Spalte nach links.<br>Befindet sich die Schreibmarke am Anfang eines Feldteils, so bewegt sie sich zum letzten Zeichen des vorhergehenden Feldteils, das ≠ dem Leerzeichen '␣' ist. Ist kein vorhergehender Feldteil vorhanden, d. h. die Schreibmarke befindet sich am Anfang des 1. Feldteils, dann bewegt sich die Schreibmarke an den Anfang des vorhergehenden Bildschirmfeldes. |

→
CTRL L

bewegt die Schreibmarke um eine Spalte nach rechts.

Befindet sich die Schreibmarke am Ende eines Feldteils, so bewegt sie sich an den Anfang des nächsten Feldteils. Ist kein nächster Feldteil vorhanden, d. h. die Schreibmarke befindet sich am Ende des letzten Feldteils, so geschieht nichts.

↑
CTRL B

bewegt die Schreibmarke in die aktuelle Spalte des darüberliegenden Feldteils.

Befindet sich die Schreibmarke im 1. Feldteil, so bewegt sie sich an den Anfang des vorhergehenden Bildschirmfeldes. Ist kein vorhergehendes Bildschirmfeld vorhanden, d.h. die Schreibmarke befindet sich bereits im 1. Bildschirmfeld des Formats, so bewegt sie sich an den Anfang dieses Bildschirmfeldes.

↓
CTRL F

bewegt die Schreibmarke in die aktuelle Spalte des darunterliegenden Feldteils.

Befindet sich die Schreibmarke im letzten Feldteil, so bewegt sie sich an den Anfang des nächsten Bildschirmfeldes. Ist kein nächstes Bildschirmfeld vorhanden, d. h. die Schreibmarke befindet sich bereits im letzten Bildschirmfeld des Formats, so bewegt sie sich an den Anfang dieses Bildschirmfeldes.

⇥

Tabulatortaste; Wirkung abhängig vom Modus:

*Insert-Modus:* fügt ein Tabulator-Zeichen ein und bewegt die Schreibmarke zum nächsten Tabulatorstop.

*Overwrite-Modus:* bewegt die Schreibmarke zum nächsten Tabulatorstop.

Als Tabulatorposition gilt jeweils das 9., 17., 25.,... Zeichen (Schrittweite 8).

F15
CTRL A

wechselt zwischen Overwrite- und Insert-Modus. Standardmäßig ist der Overwrite-Modus eingeschaltet.

CHAR DELETE
CTRL X

löscht das Zeichen, auf dem die Schreibmarke steht.

WORD DELETE
CTRL D

löscht alle Zeichen ab Schreibmarken-Position bis zum Ende des mehrzeiligen Feldes.

CTRL N

trennt die Zeile ab der aktuellen Spaltenposition, wobei der Rest in den nächsten Feldteil eingefügt wird. Im Overwrite-Modus wird dabei das Zeichen, an der die Schreibmarke steht, gelöscht.

### 4.4.5 Abweichungen in der Bedienung bei den Datentypen TEXT und BYTE

Beim Arbeiten mit Bildschirmfeldern, die den INFORMIX-ONLINE-Datentypen TEXT und BYTE zugeordnet sind, gelten gegenüber dem allgemeinen Vorgehen folgende Abweichungen:

*Suchen*

Da über das Format keine Eingaben in TEXT- oder BYTE-Felder möglich sind, können Sie in solchen Feldern auch nicht mit Wertvorgaben oder Suchoperatoren suchen.

*Anzeige von Feldinhalten*

*Datentyp TEXT:* Im Format ist die Anzeige des Inhalts nur über ein mehrzeiliges Bildschirmfeld möglich. Ein mehrzeiliges Bildschirmfeld vereinbaren Sie im Formatprogramm mit dem Attribut WORDWRAP (siehe Kapitel 3.3).
Wenn Sie die Menüfunktion *Blob* verwenden, können Sie sich den Feldinhalt über das zugeordnete Programm (z. B. Editor) anzeigen lassen (*Programm zuordnen* siehe unten).

*Datentyp BYTE*: Im Format ist die Anzeige des Inhalts nicht möglich. Dort sind BYTE-Felder lediglich mit der Anzeige <BYTE Wert> markiert.
Die Anzeige des tatsächlichen Inhalts ist nur über die Menüfunktion *Blob* möglich, über die Sie das zugeordnete Programm (siehe unten) aufrufen.

*Eingaben*

Direkte Eingaben in Bildschirmfelder sind nicht möglich. Wenn Sie nach Aufruf der Menüfunktion *Neuaufnehmen* oder *Korrigieren* die Schreibmarke an ein TEXT- oder BYTE-Feld positionieren und dort ein Ausrufezeichen '!' eingeben, wird das zugeordnete Programm (siehe unten) aktiviert. Über dieses Programm bearbeiten Sie dann den Inhalt eines solchen Feldes. Nach Programmende erhalten Sie wieder das entsprechende PERFORM-Menü. Dort entscheiden Sie, ob die über das Programm durchgeführte Eingabe in der Datenbank gespeichert werden soll [START] oder nicht [DEL].

Wenn einem TEXT- oder BYTE-Feld kein Programm zugeordnet ist, kann dessen Inhalt nicht bearbeitet werden (*Neuaufnehmen* oder *Korrigieren*).

*Programm zuordnen*

*Datentyp TEXT:* ein Programm ordnen Sie im Formatprogramm über das Attribut PROGRAM zu (siehe Kapitel 3.3). Ohne PROGRAM gilt der in der Umgebungsvariablen DBEDIT vereinbarte Editor. Ist DBEDIT nicht gesetzt, gilt der Editor des Betriebssystems.

*Datentyp BYTE:* ein Programm ordnen Sie im Formatprogramm über das Attribut PROGRAM zu. Ohne PROGRAM kann BYTE nicht bearbeitet werden.

## 4.5 PERFORM-Menüfunktionen

Die Menüfunktionen sind in der ersten Zeile des Menübereichs angezeigt. Da sie nicht alle auf eine Bildschirmzeile passen, sind sie auf zwei nebeneinanderliegende Bildschirmzeilen aufgeteilt (siehe Abschnitt 4.3 *Der Menübereich*).

### Eine Menüfunktion auswählen

Eine Menüfunktion wählen Sie aus, indem Sie entweder

- mit der Leertaste die Schreibmarke zur gewünschten Funktion bewegen und dann auf [↵] drücken oder
- den ersten Buchstaben der gewünschte Funktion eingeben, z.B. [S] für *Suchen*

### Sonderfunktionen

Erklärungen zum Arbeiten mit PERFORM:

Mit Hilfe der Taste [HELP] erhalten Sie am Bildschirm Hilfetexte zur Bedienung von PERFORM.

Beim Suchen alle bzw. eine begrenzte Anzahl von Sätzen anzeigen lassen:

Nachdem man die Funktion *Suchen* ausgewählt hat, kann man entweder

- auf Taste [START] drücken, um alle Sätze anzeigen zu lassen oder
- zuerst Wertvorgaben eingeben, die die Suche begrenzen, und dann auf diese Taste drücken (siehe Abschnitt 4.5.1 *Suchen*).

Einen korrigierten oder neueingegebenen Satz in die Datenbank aufnehmen:

Nachdem man einen neuen Satz eingegeben oder einen bestehenden Satz korrigiert hat, drückt man auf Taste [START], um den neuen bzw. korrigierten Satz aufzunehmen.

Eine bereits ausgewählte Funktion verwerfen:

bei *Suchen, Neuaufnehmen, Korrigieren, Loeschen, PRINT*:
Nachdem der Benutzer bereits eine dieser Funktionen ausgewählt hat, kann er den Vorgang mit Taste [DEL] rückgängig machen, d.h. er kehrt zum PERFORM-Menü zurück.

Kommandos auf Betriebssystemebene eingeben:

Sie können vom FORMAT-Menü aus Betriebssystemkommandos ablaufen lassen, indem Sie ein Ausrufezeichen ! eingeben. Das Ausrufezeichen wird am unteren Bildschirmrand angezeigt:

- wenn Sie Ausrufezeichen ! und dann ein Betriebssystemkommando eingegeben haben, erscheint nach Ausführung des angegebenen Kommandos die Meldung *Zur Fortsetzung bitte* ↵. Mit der Taste [↵] kehren Sie in das FORMAT-Menü zurück.
- wenn Sie Ausrufezeichen ! und dann sh eingegeben haben, wird eine neue Shell eröffnet. Sie können dann nur durch Drücken der Taste [END] in das FORMAT-Menü zurückkehren.

### Aktuelle Tabelle

Obwohl ein Format Spalten aus mehreren Tabellen enthalten kann, beziehen sich die PERFORM-Funktionen immer nur auf eine Tabelle. Diese Tabelle heißt die aktuelle Tabelle und wird von PERFORM in der zweiten Menüzeile angezeigt (z.B. Tabelle 1: kunde).

### Aktueller Satz

Der aktuelle Satz ist der Satz, der am Bildschirm angezeigt wird. Mit den Funktionen *Rueckw.* oder *Vorw.* kann man einen anderen Satz zum aktuellen Satz machen.

### Aktuelle Liste

Manche Funktionen beziehen sich auf eine *aktuelle Liste*, die durch jeden Suchvorgang erzeugt wird. Die aktuelle Liste besteht aus allen Sätzen, die die Suchbedingungen erfüllen (siehe Abschnitt 4.5.1, *Suchen*).

Die Funktionen *Vorw.*, *Rueckw.*, *Loeschen*, *PRINT*und *Korrigieren* wirken erst, wenn eine aktuelle Liste erzeugt wurde.

### 4.5.1 Suchen

Wenn Sie ein Format aufrufen, ist die erste Menüfunktion *Suchen* schon markiert.

Sie können entweder

- nach allen Sätzen der Tabelle, oder
- nur nach bestimmten Sätzen suchen.

PERFORM löscht mit Ausnahme von Join-Feldern ohne QUERYCLEAR automatisch alle Bildschirmfelder der aktuellen Tabelle im Format und positioniert die Schreibmarke auf das erste Feld.

*Einschränkung für die INFORMIX-ONLINE-Datentypen TEXT- und BYTE*

Da in diese Felder über das Format keine Eingaben möglich sind, ist in diesen Feldern auch kein Suchen mit Wertvorgaben oder Suchoperatoren möglich.

**1. Nach allen Sätzen suchen:**

Nachdem Sie auf [START] gedrückt haben, erzeugt PERFORM eine aktuelle Liste, die alle Sätze aus der aktuellen Tabelle enthält, und zeigt den ersten Satz dieser Liste im Arbeitsbereich an.

In der Statuszeile erscheint die Meldung:

```
Gefundene Saetze: n
```

n ist die Anzahl der gefundenen Sätze.

Sie können mit *Vorw.* oder *Rueckw.* vorwärts bzw. rückwärts durch die aktuelle Liste blättern.

Um zum FORMAT-Menü zurückzukehren, drücken Sie [END].Wenn Sie PERFORM auf Betriebssystemebene aufgerufen haben, wird jetzt PERFORM beendet und Sie erhalten die Kommandozeile des Betriebssystems.

**2. Nach bestimmten Sätzen suchen:**

Wenn Sie nach bestimmten Sätzen suchen, können Sie entweder

- mit einer konkreten Wertvorgabe (z.B. Muenchen) oder
- mit einem Suchoperator (z.B. >, <, =)

arbeiten.

**Suchen mit Wertvorgaben** (Selektion)

Eine Wertvorgabe kann je nach Datentyp des Feldes, in dem Sie suchen,

- ein Wort (CHAR, VARCHAR) oder
- eine Zahl (INTEGER, SMALLFLOAT, usw.)

sein.

Um mit Wertvorgaben zu Suchen, gehen Sie folgendermaßen vor:

- nachdem Sie die Menüfunktion *Suchen* ausgewählt haben, bewegen Sie die Schreibmarke mit [↵] zu dem Feld, in das Sie die Wertvorgabe schreiben wollen
- geben Sie den konkreten Wert, nach dem Sie suchen, ein und
- schicken Sie Ihre Suche mit Wertvorgabe mit [START] ab

PERFORM durchsucht die Tabelle und erzeugt eine aktuelle Liste. Diese aktuelle Liste enthält alle Sätze, die die Suchbedingung (Wertvorgabe) erfüllen.

In der Statuszeile erscheint die Meldung:

```
Gefundene Saetze: n
```

n ist die Anzahl der gefundenen Sätze.

**Beispiel**

Sie suchen im Format *muster* alle Kunden, die in München wohnen. Nachdem Sie die Menüfunktion *Suchen* ausgewählt haben, gehen Sie in das Feld *Ort*. Hier geben Sie *Muenchen* ein. Anschließend drücken Sie auf START.

Die durch das Suchen erzeugte aktuelle Liste enthält alle Sätze, in denen *Ort* den Wert *Muenchen* enthält.

Falls Sie eine Wertvorgabe eingegeben haben, der kein Satz entspricht, erscheint in der Statuszeile die folgende Meldung:

```
Es gibt keine Saetze, welche die Bedingungen erfuellen
```

**Suchen mit Suchoperatoren**

Um mit Suchoperatoren zu suchen, gehen Sie folgendermaßen vor:

- nachdem Sie die Menüfunktion *Suchen* ausgewählt haben, bewegen Sie die Schreibmarke mit ↵ zu dem Feld, in das Sie schreiben wollen
- geben dann den Wertebereich, nach dem Sie suchen, ein und
- schicken Ihre Suche mit START ab.

PERFORM durchsucht die Tabelle und erzeugt eine aktuelle Liste. Diese aktuelle Liste enthält alle Sätze, die die Suchbedingung erfüllen.

In der Statuszeile erscheint die Meldung:

```
Gefundene Saetze: n
```

n ist die Anzahl der gefundenen Sätze.

**Beispiel**

Sie suchen im Format *muster* alle Kunden, die Waren für mehr als DM 500,00 bestellt haben. Nachdem Sie die Menüfunktion *Suchen* ausgewählt haben, gehen Sie in das Feld *Gesamtpreis*. Hier geben Sie >500 ein. Anschließend drücken Sie auf START.

Die durch das Suchen erzeugte aktuelle Liste enthält alle Sätze, in denen Gesamtpreis >500 ist.

In der Statuszeile erscheint die Meldung:

```
Gefundene Saetze: 8
```

Falls Sie eine Wertvorgabe eingegeben haben, der kein Satz entspricht, erscheint in der Statuszeile die folgende Meldung:

```
Es gibt keine Saetze, welche die Bedingungen erfuellen
```

Folgende Suchoperatoren stehen Ihnen zur Verfügung:

| Suchoperator | Bedeutung | Datentyp | Syntax |
|---|---|---|---|
| = | gleich | alle | =x |
| > | größer als | alle | >x |
| < | kleiner als | alle | <x |
| >= | größer-gleich | alle | >=x |
| <= | kleiner-gleich | alle | <=x |
| <> od. != | ungleich | alle | <>x od. !=x |
| \| | oder | alle außer VAR[CHAR] | x\|y |
| : | zwischen | alle | x:y |
| .. | zwischen | DATETIME, INTERVAL | x..y |
| ? | Joker (n=1) | VAR[CHAR] | ?x, x?, ?x? |
| * | Joker (n>=1) | VAR[CHAR] | *x, x*, *x* |
| >> | höchster Wert | alle | >> |
| << | niedrigster Wert | alle | << |

**Erklärung**

= (mit Wert): entspricht dem Suchen mit Wertvorgabe (siehe oben).

= (ohne Wert): PERFORM sucht nach Sätzen, die in diesem Feld leer sind.

=* PERFORM sucht nach Sätzen, die in diesem Feld Zeichen * als Wert enthalten.

x Stellt jedes Zeichen dar, das zu dem Datentyp paßt. Schreiben Sie den Wert unmittelbar, d.h. ohne Leerstelle, nach dem Suchoperator.

> Bei VAR[CHAR]-Feldern: ein Zeichen mit einer höheren ASCII-Wert. (ASCII-Tabelle, siehe Anhang A.3)
> Bei DATE[TIME]-Feldern: ein jüngeres Datum.

< Bei VAR[CHAR]-Feldern: ein Zeichen mit einer niedrigeren ASCII-Nummer. Bei DATE[TIME]-Feldern: ein älteres Datum.

| Stellt eine OR-Bedingung dar. Die Angabe kann für Spalten des Typs CHAR oder VARCHAR nicht verwendet werden. Die Eingabe '|' in solche Felder wird als 'ö' interpretiert, das denselben ASCII-Code belegt.

*Beispiel:* Die Eingabe 112|118|116 im Feld Kundennummer bedeutet Kundennummer 112 oder 118 oder 116.

: Bezeichnet einen Wertebereich in der Form: a : b, wobei a kleiner als b sein muß. b gilt inklusive. Zum Beispiel bedeutet die Angabe 1:4 'es gelten die Zahlen 1 bis einschließlich 4'.
Enthält ein Suchbegriff eine ungerade Anzahl Doppelpunkte, wird automatisch der mittlere Doppelpunkt als Suchoperator ausgewertet. Bei gerader Anzahl von Doppelpunkten erscheint eine Fehlermeldung, wenn der Suchbegriff nicht zufällig ein gültiger DATETIME- oder INTERVAL-Wert ist.

.. gleiche Bedeutung wie Doppelpunkt. Diesen Suchoperator verwenden Sie für die Datentypen DATETIME und INTERVAL zur besseren Kennzeichnung des Suchoperators.

? Ein Joker-Zeichen, das für ein beliebiges Zeichen steht.

*Beispiel*
'?etzen' könnte Aetzen, Fetzen, hetzen, usw. bedeuten.

* Ein Joker-Zeichen, das für kein oder eine beliebige Anzahl von Zeichen steht.

*Beispiel*
'M*' könnte M, München, Mannheim, M1, M999 usw. bedeuten.

>> Findet den höchsten Wert für ein bestimmtes Feld.

<< Findet den niedrigsten Wert für ein bestimmtes Feld.

### 4.5.2 Vorw. und Rueckw.

Sie können diese Funktionen erst dann aktivieren, wenn Sie schon eine aktuelle Liste erzeugt haben (siehe die Einleitung zu diesem Abschnitt und Abschnitt 4.5.1, *Suchen*).

Nachdem Sie die aktuelle Liste erzeugt haben, sehen Sie nur den ersten Satz am Bildschirm. Mit *Vorw.* oder *v* können Sie zum nächsten Satz springen, oder Sie können z.B. um 3 Sätze mit '3' und *Vorw.* (oder 3v) springen.

Mit *Rueckw* können Sie zum vorhergehenden Satz springen, oder Sie können z.B. um 3 Sätze rückwärts mit '3' und *Rueckw.* springen (oder 3r).

Falls Sie beim letzen bzw. ersten Satz versuchen, noch weiter zu blättern, erhalten Sie die folgende Meldung:

```
In dieser Richtung ist die aktuelle Liste zu Ende
```

### 4.5.3 Blob

Mit dieser Funktion können Sie sich den Inhalt von TEXT- und BYTE-Feldern anzeigen lassen. Da diese Datentypen nur dem Anwender von INFORMIX-ONLINE zur Verfügung stehen, ist diese Funktion nur unter INFORMIX-ONLINE von Bedeutung.

*Ablauf*

Nachdem Sie ein Format aufgerufen haben,

- wählen Sie *Suchen* und [START], um eine aktuelle Liste zu erstellen
- wählen Sie die *Blob*-Funktion

Sie erhalten folgendes Menü:

```
BLOB:  START beendet.  Pfeiltasten bewegen zu BLOBs.  ' !' zeigt BLOB.
```

Die Schreibmarke wird jetzt in das erste BLOB-Feld (TEXT oder BYTE) des Formats bewegt, dem ein Programm zugeordnet ist.

Eine Programmzuordnung besteht in folgenden Fällen:

*Datentyp TEXT:* immer. Als Programm gilt das im PROGRAM-Attribut des Formatprogramms vereinbarte Programm (siehe Kapitel 3.3). Fehlt die Angabe PROGRAM, so gilt der Editor, der in der Umgebungsvariablen DBEDIT (siehe 7.2) vereinbart ist. Ist DBEDIT nicht gesetzt, gilt der Standardeditor des Systems.

*Datentype BYTE:* nur, wenn über das PROGRAM-Attribut ein Programm vereinbart wird. Fehlt PROGRAM, gibt es keine Zuordnung.

Mit der Taste [↵] wechseln Sie jeweils zum nächsten BLOB-Feld, wobei BLOB-Felder ohne Programmzuordnung übersprungen werden.

Wenn Sie in dem BLOB-Feld, in dem sich die Schreibmarke gerade befindet, ein Ausrufezeichen '!' eingeben, wird das zugeordnet Programm aktiviert. INFORMIX kopiert den BLOB-Inhalt in eine temporäre Datei auf Platte und stellt diese Datei dem Programm zur Verfügung. Die Datei wird bei Programmende wieder gelöscht. Das heißt, falls Sie über das Programm Änderungen vorgenommen haben, werden diese Änderungen nicht berücksichtigt. Nach Programmende erscheint wieder das FORMAT-Menü.

Die Funktion *Blob* dient lediglich zur Anzeige von BLOB-Daten. Zum Ändern oder Neuaufnehmen von BLOB-Daten benutzen Sie die Funktionen *Korrigieren* bzw. *Neuaufnehmen.* Im Unterschied zu *Blob* erscheint bei diesen Funktionen nach Programmende das Menü, über das Sie den Inhalt in die Datenbank einspeichern können.

Beim Datentyp TEXT haben Sie zusätzlich die Möglichkeit, sich Inhalte über WORDWRAP anzeigen zu lassen (näheres siehe Kapitel 3.3, ATTRIBUTES-Abschnitt).

### 4.5.4 Neuaufnehmen

Mit dieser Funktion nehmen Sie neue Sätze in eine Tabelle auf. Die Eingabe 3n bewirkt, daß die Funktion *Neuaufnehmen* dreimal erscheint. Falls Sie dann weniger als die angebene Anzahl von neuen Sätzen aufnehmen wollen, drücken Sie auf die Taste DEL.

Die Werte geben Sie direkt in die entsprechenden Bildschirmfelder ein.

*Ausnahmen*

- In Felder des Datentyps TEXT und BYTE können über das Format keine Eingaben gemacht werden. Eingaben sind nur über ein eigens dafür vorgesehenes Programm möglich (näheres siehe Abschnitt 4.4.5).
- In Felder, für die im ATTRIBUTES-Abschnitt NOENTRY (siehe Kapitel 3) vereinbart wurde, können Sie keinen Wert eingeben.

Wenn Sie fertig sind,

- drücken Sie auf START
  die Eingabe wird aufgenommen, d.h. die Tabelle enthält nun einen Satz mehr; oder
- drücken Sie auf DEL
  die Eingabe wird nicht aufgenommen.

(Diese Aufforderungen stehen während des Neuaufnehmens in der ersten Bildschirmzeile.)

Wenn Sie auf die auf die Taste START gedrückt haben, erscheint im Meldungsbereich die Meldung:

```
Der Satz wurde neu aufgenommen
```

### 4.5.5 Korrigieren

Mit dieser Funktion können Sie einen Satz in einer Tabelle ändern.

Nachdem Sie einen Satz ausgesucht haben (siehe Abschnitt 4.5.1, *Suchen*), können Sie die Daten in einem beliebigen Feld ändern, indem Sie sie einfach überschreiben.

*Ausnahmen*

- Felder des Datentyps TEXT und BYTE lassen sich nicht überschreiben. Eingaben sind nur über ein eigens dafür vorgesehenes Programm möglich (näheres siehe Abschnitt 4.4.5).
- Felder, für die im ATTRIBUTES-Abschnitt NOUPDATE (siehe Kapitel 3.3) vereinbart wurde, lassen sich nicht ändern.
- Felder, die über Join mit Feldern verbunden sind, für die das Attribut VERIFY (Eingabe wiederholen) vereinbart ist.
  In solchen Fällen muß zuerst der Feldinhalt des Feldes geändert werden, für das VERIFY gilt, bevor das zugeordnete Join-Feld (ohne VERIFY) geändert wird.

Wenn Sie mit dem Korrigieren fertig sind,

- drücken Sie auf [START]
  die Änderungen werden angenommen oder
- drücken Sie auf [DEL]
  die Änderungen werden ignoriert

(Diese Aufforderungen stehen während des Korrigierens in der ersten Bildschirmzeile.)

Wenn Sie die Taste [START] gedrückt haben, erscheint im Meldungsbereich die folgende Meldung:

```
Der Satz wurde geaendert
```

Dieser Satz erscheint auch dann, wenn Sie überhaupt nichts geändert, aber [START] gedrückt haben.

### 4.5.6 Loeschen

Diese Funktion erlaubt das Löschen eines Satzes einer Tabelle. Nachdem Sie einen Satz ausgesucht haben, den Sie löschen wollen (siehe Abschnitt 4.5.1, *Suchen*), wählen Sie *Loeschen.* PERFORM fordert Sie jetzt auf, den Löschvorgang zu bestätigen. Wählen Sie entweder *Ja* oder *Nein.* Wenn Sie *Ja* wählen, zeigt PERFORM die folgende Meldung:

```
Der Satz wurde geloescht
```

Ein Satz, der ein überprüfendes Join Feld enthält, kann in der dominanten Tabelle nur gelöscht werden, wenn der Satz zuvor in allen verbundenen Tabellen gelöscht wurde (siehe Kapitel 3.3).

### 4.5.7 Tabelle

Enthält ein Format Felder aus mehreren Tabellen, lassen sich immer nur die Felder der zugeordneten aktuellen Tabelle bearbeiten. Im Menübereich wird die momentan aktuelle Tabelle angezeigt.

Mit der Funktion *Tabelle* aktivieren Sie jeweils die nächste Tabelle des Formats. Dabei entspricht die Reihenfolge, in der die Tabellen aktiviert werden, der Reihenfolge, in der die Feldbezeichner (einschließlich Join) im ATTRIBUTES-Abschnitt des Formatprogramms definiert sind.

Mit der Angabe *anzahl* [T] können Sie auch direkt in die angegebene Tabelle wechseln. Zum Beispiel aktiviert 3 [T] die dritte Tabelle des Formats.

PERFORM gibt für die aktivierte Tabelle immer die Bildschirmseite zuerst aus, die die meisten Felder dieser Tabelle enthält.
Die aktivierten Felder werden mit Feldbegrenzer dargestellt. Bei nicht akivierten Feldern fehlen die Feldbegrenzer.

### 4.5.8 Format

Ist für das Format mehr als 1 Bildschirmseite definiert (siehe SCREEN-Abschnitt des Formatprogrammes, so können Sie die nächste Bildschirmseite ausgeben, indem Sie

- *Format* auswählen oder
- auf die Taste [F] oder
- auf die Taste [⇩] drücken.

Jeweils die vorige Seite können Sie sehen, indem Sie auf die Taste [⇧] drücken.

Nach der letzen Seite erhalten Sie automatisch die 1. Bildschirmseite.

### 4.5.9 Aktuell

Diese Funktion hat drei verschiedene Anwendungen:

1. Wenn ein Format ein Join-Feld enthält, hat jede Tabelle, aus der Daten am Bildschirm angezeigt werden, eine eigene aktuelle Liste. Während Sie Daten aus einer anderen Tabelle lesen, kann es vorkommen, daß Sie vergessen, welche die erste aktive Liste ist. Die Funktion *Aktuell* (oder A) bringt Sie zum ersten aktuellen Satz zurück.

2. Bei Mehrbenutzerbetrieb kann es passieren, daß ein anderer Benutzer einen Satz korrigiert, während Sie den selben Satz lesen. *Aktuell* liest den Satz neu ein und zeigt ihn am Bildschirm als die letzte Information.

3. Ebenfalls bei Mehrbenutzerbetrieb kann Ihr Bildschirm durch eine Meldung von einem anderen Benutzer, oder vom System, zerstört werden. *Aktuell* baut Ihren Bildschirm neu auf. Dieselbe Funktion erfüllt auch die Tastenfolge CTRL R.

### 4.5.10 Master und Detail

Wenn im INSTRUCTIONS-Abschnitt eines Formatprogramms (siehe Abschnitt 3.3) eine MASTER/DETAIL-Beziehung zwischen den verschiedenen, in diesem Programm genannten Tabellen definiert wurde, ermöglichen die Funktionen *Master* (oder [M]) und *Detail* (oder [D]) das Hin- und Herschalten zwischen den beiden Tabellenarten.

Wenn Sie *Detail* aktivieren, nimmt PERFORM die aktuellen Werte in den Join-Feldern als Suchbedingungen und sucht in der Detail-Tabelle, erzeugt eine aktuelle Liste von Sätzen, die diese Bedingungen erfüllen, und zeigt den ersten Satz dieser Liste am Bildschirm an.

Sind mehrere Detail-Tabellen für eine Master-Tabelle definiert, dann erhalten Sie mit der Angabe *Detail* die 1. Detailtabelle, mit der Angabe *D nr* die Detailtabelle *nr* (nr ist eine Ziffer).

*Hinweis*

Hat eine DETAIL-Tabelle mehrere Seiten, wird diejenige Seite angesprungen, die die meisten Bildschirmfelder enthält.

Wenn Sie jetzt *Master* aktivieren, kehren Sie zur Master-Tabelle zurück, ohne daß eine neue Suche ausgeführt wird.

Wenn im Formatprogramm keine MASTER/DETAIL-Beziehung definiert wurde, erscheint nach Aktivierung der Funktion *Master* die Fehlermeldung:

```
Fuer diese Tabelle wurde keine MASTER-Tabelle definiert
```

### 4.5.11 PRINT

PRINT gibt den aktuellen Satz oder die gesamte aktuelle Liste (siehe *aktuelle Liste*) in eine Datei aus.
Das Ausgabeformat entpricht entweder der Bildschirmdarstellung oder dem Format, das auch die SQL-Anweisung UNLOAD erzeugt.

*Ablauf*

Nachdem Sie ein Format aufgerufen haben,

- wählen Sie *Suchen* und *Start*, um eine aktuelle Liste zu erstellen
- wählen Sie die PRINT-Funktion

Sie erhalten folgende Abfrage:

```
Ausgabeprogramm? (STD: perform.out):
```

Hier müssen Sie einen Datei vereinbaren, in die die Ausgabe geschrieben werden soll. Sie können sowohl einen neuen als auch einen bereits vorhandenen Dateinamen angeben. Wenn Sie hier nichts angeben, benutzt PERFORM automatisch die Datei *perform.out.*

Ist die Ausgabedatei definiert, fragt PERFORM, ob die Datei ergänzt oder überschrieben werden soll:

```
FORMAT AUSGABEPROGRAMM: Anhaengen Erstellen
```

Anhaengen
Die Ausgabe soll ans Ende der Datei geschrieben werden.

Erstellen
Die Ausgabedatei soll überschrieben werden.

Dann vereinbaren Sie das Ausgabeformat:

```
AUSGABE FORMAT: ASCII-Format  Bildschirm-Format
```

ASCII-Format
Das Ausgabeformat entspricht dem Format, das die UNLOAD-Anweisung (siehe SQL-Handbuch [1]) erzeugt. Je Ausgabesatz wird ein (logischer) Satz in die Datei geschrieben.
Die einzelnen Feldinhalte werden durch das Standard-Trennzeichen (gemäß Umgebungsvariable DBDELIMITER, siehe Kapitel 7.2) getrennt. Die Reihenfolge der Feldinhalte entspricht der Reihenfolge im ATTRIBUTES-Abschnitt des Formatprogramms. Ausnahme: ein LOOKUP-Feld wird an das Satzende angehängt.
Bei Formaten, die aus mehreren Bildschirmen bestehen, wird der Inhalt aller Bildschirme berücksichtigt. Sätze in diesem Format können mit der READ-Anweisung des ACE (siehe Kapitel 5.3) und mit der LOAD-Anweisung (siehe SQL-Handbuch [1]) bearbeitet werden.

Bildschirm-Format
Das Ausgabeformat entspricht der Bildschirmdarstellung. Jeder Satz wird so ausgegeben, wie er am Bildschirm dargestellt ist. Besteht ein Format aus mehreren Bildschirmen, so wird nur der aktuelle Bildschirm ausgegeben. Sie müssen dann den gesamten Ausgabe-Vorgang für jeden Bildschirm wiederholen.

```
FORMAT-AUSGABE:  Gesamt  Aktuell
```

Gesamt
für die ganze Liste, oder
Aktuell
für den aktuellen (am Bildschirm angezeigten) Satz.

### 4.5.12 END

Mit dieser Funktion beenden Sie PERFORM und kehren zum FORMAT-Menü bzw. zur Kommandozeile des Betriebssystems zurück.

Die Funktion können Sie eingeben, indem Sie

- die Schreibmarke auf END setzen und auf [↵] drücken oder
- auf [E] drücken oder
- auf [END] drücken

## 4.6 Besonderheiten bei der PERFORM-Bedienung

Ein Format basiert auf einem ablauffähigen Formatprogramm (siehe Kapitel 3). In diesem Programm wurden Festlegungen bezüglich der Ein- und Ausgabe von Werte getroffen. Diese Festlegungen wirken sich auf die Arbeit mit PERFORM aus.

Es wurde zum Beispiel ein Bildschirmfeld einer Tabellenspalte zugeordnet. Diese Tabellenspalte hat einen bestimmten Datentyp. Der Benutzer kann dann in das zugeordnete Bildschirmfeld nur noch Werte dieses Datentyps eingeben oder ausgeben lassen. Oder es wurde im Formatprogramm für ein Bildschirmfeld über DEFAULT ein Standardwert definiert. Dieser Wert erscheint dann im Bildschirmfeld, noch bevor der Benutzer Eingaben machen kann.

Die folgende Aufstellung soll Ihnen eine Hilfestellung dafür geben, einige Besonderheiten bei der PERFORM-Bedienung mit Anweisungen, die im Formatprogramm oder beim Erzeugen einer Tabelle angegeben wurden, in Zusammenhang zu bringen.

| **PERFORM-Verhalten** | **Ursache** |
|---|---|
| Werte werden nicht nicht angenommen. Fehlermeldung: "Fehler im Feld". | In ein Bildschirmfeld dürfen Sie nur Werte desjenigen Datentyps eingeben, auf den dieses Feld festgelegt wurde. Diese Festlegung wurde bereits beim Erzeugen der Tabelle getroffen (SQL-Anweisung CREATE TABLE, siehe SQL-Handbuch [1]). Eine ausführliche Erläuterung zu den einzelnen Datentypen finden Sie im SQL-Handbuch [1]. |
| Bewegung der der Schreibmarke ist unklar. | Sofern keine spezielle andere Anweisung im ATTRIBUTES- oder INSTRUCTIONS-Abschnitt des Formatprogramms definiert wurde, bewegt sich die Schreibmarke automatisch in der Reihenfolge, in der die Feldbezeichner im ATTRIBUTES-Abschnitt des Formatprogramms angegeben wurden. Diese Reihen- |

| | |
|---|---|
| | folge kann verändert werden, indem entweder<br>– die Reihenfolge, in der die Feldbezeichner im ATTRIBUTES-Abschnitt des Formatprogramms erscheinen, verändert wird (siehe Kapitel 3) oder<br>– im INSTRUCTIONS-Abschnitt des Formatprogramms die Anweisung NEXTFIELD definiert wird (siehe Kapitel 3). |
| Buchstaben werden anders dargestellt als sie eingegeben werden. | Es wurde UPSHIFT oder DOWNSHIFT definiert (siehe Kapitel 3.3, *Attribute*). |
| Werte vom Datentyp FLOAT, SMALLFLOAT und DATE werden anders dargestellt als sie eingegeben werden. | Über FORMAT wurde die die Anzahl der Stellen links und rechts vom Komma definiert (siehe Kapitel 3.3, *Attribute*). |
| PERFORM gibt die Meldung "Diese Eingabe ist nicht erlaubt" aus. | Durch INCLUDE wurde ein Wertebereich festgelegt (siehe Kapitel 3.3, *Attribute*). |
| Sie erhalten ein akustisches Signal und die Eingabe wird nicht angenommen. | Durch PICTURE wurde die Eigabe auf ein bestimmtes Muster eingeschränkt (siehe Kapitel 3.3, *Attribute*). |
| Sie erhalten die Meldung "In dieses Feld muss eine Eingabe gemacht werden." | Es wurde REQUIRED definiert (siehe Kapitel 3.3, *Attribute*). |
| Sie erhalten die Meldung "Eingabe zur Kontrolle wiederholen". | Es wurde VERIFY definiert (siehe Kapitel 3.3, *Attribute*). |

| | |
|---|---|
| Die Schreibmarke geht über ein Bildschirm-Feld hinweg. | Falls der Benutzer in der aktuellen Tabelle arbeitet und das Bildschirmfeld weder ein SERIAL-Feld noch ein LOOKUP-Feld ist, wurde NOENTRY, NOUPDATE oder die Anweisung NEXTFIELD definiert (siehe Kapitel 3.3, *Attribute* bzw. *Anweisungen für Kontrollblöcke*). |
| Die Schreibmarke geht nach der Eingabe automatisch in ein anderes Bildschirmfeld. | Es wurde AUTONEXT oder die Anweisung NEXTFIELD definiert (siehe Kapitel 3.3, *Attribute* bzw. *Anweisungen für Kontrollblöcke*). |
| Werte werden nach rechts gerückt. | Es wurde RIGHT definiert (siehe Kapitel 3.3, *Attribute*). |
| Numerische Werte werden nach rechts gerückt und links mit Nullen aufgefüllt. | Es wurde ZEROFILL definiert (siehe Kapitel 3.3, *Attribute*). |
| Ein Wert, der nicht eingegeben wurde, erscheint im Bildschirm-Feld. | Es wurde DEFAULT, PICTURE, ein Join oder die Anweisung LET definiert (siehe Kapitel 3.3, *Attribute* oder *Anweisungen für Kontrollblöcke*). |
| Sie erhalten die Meldung "Die Eingabe ist in Tabelle ..... nicht vorhanden". | Es wurde ein überprüfender Join definiert (siehe Kapitel 3.3, ATTRIBUTES-Abschnitt) |
| Es erscheint eine invers dargestellte Meldung und/oder Sie erhalten ein akustisches Signal. | Es wurde die Anweisung oder das Attribut COMMENTS definiert (siehe Kapitel 3.3, *Attribute* oder *Anweisungen für Kontrollblöcke*). |
| Daten werden gesichert bevor Sie START gedrückt haben. | Es wurde die Anweisung NEXTFIELD EXITNOW definiert (siehe Kapitel 3.3, *Anweisungen für Kontrollblöcke*). |

In die Bildschirmfelder einer bestimmten Tabelle können Werte nicht eingetragen, gesucht korrigiert oder gelöscht werden.

Durch Zugriffsrechte kann ein Benutzer daran gehindert werden, Daten in eine bestimmte Tabelle einzutragen oder in einer bestimmte Tabelle zu suchen, zu korrigieren oder zu löschen. Diese Zugriffsrechte werden über SQL-Anweisungen definiert (siehe SQL-Handbuch [1]). Über die Funktion *Info* in den Menüs SQL-Dialog oder TABELLE können Sie sich informieren, wie die Zugriffsrechte festgelegt sind.

# 5 ACE - Listen erstellen



ACE ist ein Werkzeug, mit dem Sie Listen erstellen können. Die Liste wird aus Daten erstellt, die aus einer oder mehreren Tabellen einer Datenbank kommen. Diese Datenbank und die entsprechenden Tabellen müssen schon bestehen, bevor Sie mit ACE arbeiten.

Sie legen in einem Listen-Quellprogramm fest, welche Informationen in einer Liste stehen, und wie INFORMIX sie ausgeben soll. Dieses Quellprogramm übersetzen Sie dann mit der ACE-Komponente ACEPREP. Dadurch erzeugen Sie ein ablauffähiges Listenprogramm. Dieses läuft unter der ACE-Komponente ACEGO ab und erstellt die gewünschte Liste.

Sie können alle Schritte bis hin zum Starten eines Listenprogramms entweder

- im INFORMIX-Menüsystem oder
- auf der Betriebssystemebene erledigen

Für beide Fälle ist wichtig, daß das Verzeichnis der Datenbank, für die Sie eine Liste erstellen wollen,

- ihr aktuelles Verzeichnis ist oder
- die Umgebungsvariable DBPATH mit diesem Verzeichnis belegt ist (siehe Kapitel 7.2).

## 5.1 Eine Liste im Menüsystem erstellen

Wenn Sie im Hauptmenü INFORMIX-SQL die Funktion *Liste* auswählen, erhalten Sie das Menü LISTE.

```
LISTE:   Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
```

Ablauf
: startet ein ablauffähiges Listenprogramm. Dieses erstellt eine Liste. Nach Auswahl der Funktion können Sie unter den ablauffähigen Listenprogrammen auswählen.

Modifizieren
: Sie verändern ein Listen-Quellprogramm. Nach Auswahl der Funktion können Sie unter den vorhandenen Quellprogrammen auswählen. INFORMIX lädt dann das ausgewählte Quellprogramm in den eingestellten Editor (siehe Abschnitt 5.1.4). Nachdem Sie den Editor verlassen haben, führt INFORMIX Sie solange, bis das ablauffähige Programm erzeugt ist (siehe Abschnitt 5.1.2).

Neu
: Sie schreiben ein neues Listen-Quellprogramm. Die Funktion wirkt fast genauso wie die Funktion *Modifizieren.* Der Unterschied ist, daß Sie einen neuen Namen vergeben müssen und im eingestellten Editor keine Vorgabe haben. Sie müssen das Quellprogramm vollständig neu schreiben. Nachdem Sie den Editor verlassen haben, führt INFORMIX Sie solange, bis das ablauffähige Programm erzeugt ist.

Compilieren
: übersetzt ein Listen-Quellprogramm und erzeugt damit ein ablauffähiges Listenprogramm. Sie wählen das Quellprogramm am Bildschirm aus und die Übersetzung beginnt. Wenn das Quellprogramm Fehler enthält, dann erhalten Sie das Menü COMPILIEREN LISTE (siehe Abschnitt 5.1.3).

Generieren
: erzeugt für eine Standardliste ein ablauffähiges Programm. Dafür geben Sie der Liste einen Namen und wählen die Tabelle aus. INFORMIX erzeugt automatisch das Quellprogramm und übersetzt es. Weitere Informationen finden Sie im Abschnitt 5.1.1.

Loeschen
Sie löschen das Quellprogramm und das ablauffähige Programm einer Liste. Nach Auswahl der Funktion können Sie unter den Listenprogrammen auswählen.

Wie Sie das Menüsystem bedienen, beschreibt Kapitel 1.3.

### 5.1.1 Eine Standardliste erstellen

Eine Standardliste listet alle Sätze einer Tabelle auf. Wenn kein Satz länger als 132 Zeichen ist, enthält jede Zeile genau einen Satz. Es kann dabei auch zum Zeilenüberlauf kommen. Sind die Sätze länger als 132 Zeichen, stehen alle Feldinhalte untereinander.

Die Funktion *Generieren* im Menü LISTE erzeugt automatisch ein ablauffähiges Programm für eine Standardliste. Nachdem Sie die Funktion ausgewählt haben,

- wählen Sie eine Datenbank aus, falls noch keine aktiv ist
- tragen Sie einen maximal zehn Zeichen langen Namen für die Liste ein und
- wählen Sie die Tabelle aus, für die Sie eine Standardliste haben wollen

INFORMIX erzeugt nun das ablauffähige Programm. Dabei entsteht auch das Quellprogramm für die Standardliste.

**Hinweis**

Oft erzeugt man das Programm für eine Standardliste nur, um das Quellprogramm zu erhalten. Dieses ändert man dann mit der Funktion *Modifizieren* ab.

### 5.1.2 Ein Listen-Quellprogramm abändern

Mit der Funktion *Modifizieren* im Menü LISTE ändern Sie ein Quellprogramm für eine Liste ab.

- Wählen Sie die Funktion *Modifizieren* aus.
- Wählen Sie am Bildschirm das Quellprogramm aus, das Sie abändern wollen. Anschließend erhalten Sie den eingestellten Editor (siehe Abschnitt 5.1.4). In ihm ist eine Kopie des ausgewählten Quellprogramms geladen.
- Nun tragen Sie Ihre Änderungen ein. Dabei müssen Sie die Syntax einhalten, die im Abschnitt 5.3 beschrieben ist.
- Verlassen Sie den Editor. Dabei müssen Sie Ihre Änderungen entsprechend den Regeln des Editors sichern.

Nun erhalten das Menü MODIFIZIEREN LISTE.

```
MODIFIZIEREN LISTE:   Compilieren  Sichern-und-END  Verwerfen-und-END
```

Compilieren
: übersetzt das Quellprogramm, das der Editor abliefert, und erzeugt dadurch das ablauffähige Programm. Sie können das Programm aber erst ablaufen lassen, nachdem Sie die Funktion *Sichern-und-END* aufgerufen haben. Wenn das Quellprogramm Fehler enthält, dann erhalten Sie das Menü COMPILIEREN LISTE (siehe Abschnitt 5.1.3).

Sichern-und-END
: macht die veränderte Kopie des Quellprogramms zum Original und erzeugt die Datei mit dem ablauffähigen Listenprogramm.

Verwerfen-und-END
: verwirft die veränderte Kopie des Quellprogramms und auch das übersetzte ablauffähige Programm. Das alte Quellprogramm und das alte ablauffähige Programm für die entsprechende Liste bleiben unverändert.

### 5.1.3 Fehler nach Aufruf der Funktion Compilieren

Wenn beim Übersetzen Fehler auftreten, dann erhalten Sie das Menü COMPILIEREN LISTE.

```
COMPILIEREN LISTE:   Korrigieren  END
```

Korrigieren
: ruft den eingestellten Editor (siehe Abschnitt 5.1.4) mit einer Korrekturkopie des Quellprogramms auf. In der Korrekturkopie stehen Fehlerkommentare. Sie können nun die Fehler korrigieren und dabei die Fehlerkommentare stehen lassen. Anschließend sichern Sie Ihre Korrekturen und verlassen den Editor. Sie erhalten wieder das Menü MODIFIZIEREN LISTE, wo Sie die Funktion *Compilieren* auswählen (siehe Abschnitt 5.1.2).

END
: kehrt in das vorhergehende Menü zurück.

### 5.1.4 Konventionen

**Name des Listenprogramms**

*Maximale Länge:* 10 Zeichen

*Erlaubte Zeichen:* Buchstaben, Ziffern, _ (Unterstrich). Das erste Zeichen muß ein Buchstabe sein.

*Verbotene Namen:* reservierte Wörter (siehe SQL-Handbuch [1])

**Editor**

Zur Bearbeitung Ihres Listenprogramms stellt Ihnen ACE den Editor zur Verfügung, der über die Umgebungsvariable DBEDIT definiert ist (DBEDIT siehe Kapitel 7.2). Das ist standardmäßig der *ced.*

**Dateien**

ACE erzeugt für jedes Listenprogramm zwei Betriebssystemdateien:

| | |
|---|---|
| name.*ace* | Datei, die das Listen(-Quell)programm enthält. |
| name.*arc* | Datei, die das übersetzte Listenprogramm enthält. |

name ist der von Ihnen definierte Name des Listenprogramms.

Diese Dateien müssen entweder im aktuellen Dateiverzeichnis oder in dem Dateiverzeichnis, das die Umgebungsvariable DBPATH festlegt, enthalten sein, damit ACE darauf zugreifen kann.

## 5.2 Eine Liste auf der Betriebssystemebene erstellen

Auf Betriebssystemebene erstellen Sie eine Liste folgendermaßen:

- Sie schreiben mit einem Editor das Quellprogramm für die Liste. Dabei spielt es keine Rolle, ob Sie es vollständig neu schreiben oder ob Sie ein vorhandenes Quellprogramm abändern. Sie verwenden die Syntax, wie sie im Abschnitt 5.3 beschrieben ist. Der Name des Listen-Quellprogramms muß das Suffix *.ace* haben.
- Sie übersetzen das Quellprogramm mit dem Programmaufruf *saceprep* (siehe Abschnitt 5.2.1) oder *isql* (siehe Kapitel 7.3.1). Im Programmaufruf geben Sie den Namen des Quellprogramms an; das Suffix *.ace* lassen Sie weg.
- Falls die Übersetzung erfolgreich war, hat INFORMIX ein ablauffähiges Listenprogramm erzeugt. Es hat bis auf das Suffix den gleichen Namen wie das Quellprogramm; das Suffix ist *.arc*.
- Falls bei der Übersetzung Fehler aufgetreten sind, hat INFORMIX eine Datei mit dem Suffix *.err* angelegt. Sie enthält eine Kopie des Quellprogramms mit Fehlerkommentaren. Sie korrigieren diese Datei mit einem Editor; die Fehlerkommentare brauchen nicht zu löschen. Dann überschreiben Sie die *.ace*-Datei mit der korrigierten *.err*-Datei. Anschließend übersetzen Sie das Programm erneut.
- Sie erstellen die Liste, indem Sie das ablauffähige Listenprogramm starten. Dazu benutzen Sie den Programmaufruf *sacego* (siehe Abschnitt 5.2.2) oder *isql* (siehe Kapitel 7.3.1). Im Programmaufruf geben Sie den Namen des ablauffähigen Programms an; das Suffix *.arc* lassen Sie weg.

### 5.2.1 saceprep - ein Listen-Quellprogramm übersetzen

Mit dem Programmaufruf *saceprep* übersetzen Sie ein Listen-Quellprogramm. Dieses Quellprogramm steht im aktuellen Dateiverzeichnis und sein Dateiname hat das Suffix *.ace*.

Durch das Übersetzen erzeugen Sie ein ablauffähiges Listenprogramm, dessen Dateiname das Suffix *.arc* hat. Das Programm starten Sie entweder

- mit dem Programmaufruf *sacego* (siehe Abschnitt 5.2.2) oder
- über das INFORMIX-Menüsystem (siehe Abschnitt 5.1) oder
- mit dem Programmaufruf *isql* (siehe Kapitel 7.3.1).

Sie erhalten dann als Ergebnis die Liste.

Falls bei der Übersetzung Fehler auftreten, legt *saceprep* eine Fehlerkopie des Listen-Quellprogramms an. In dieser Datei stehen Fehlerkommentare und ihr Name hat das Suffix *.err*. Wenn Sie die nötigen Korrekturen in dieser Datei vornehmen, dann müssen Sie die Datei so umbenennen, daß sie das Suffix *.ace* hat. Ansonsten löscht *saceprep* die Datei mit dem Suffix *.err* beim erfolgreichen Übersetzen.

**saceprep[␣-q][␣-ansi][␣-o␣dateiverzeichnis]␣datei**

**-q** unterdrückt alle Meldungen am Bildschirm, die keine Fehlermeldungen sind.

**-ansi**
prüft, ob die SQL-Anweisungen im SELECT-Abschnitt Erweiterungen gegenüber dem ANSI-Standard enthalten. ACEPREP gibt dann eine Warnung aus.

**-o** dateiverzeichnis
schreibt die Dateien, die *saceprep* erzeugt, ins angegebene Dateiverzeichnis.

Standardmäßig kommen die Dateien in das Dateiverzeichnis, in dem das Quellprogramm steht.

datei
ist der Name der Datei, die das Listen-Quellprogramm enthält. Er muß das Suffix *.ace* haben, aber Sie brauchen das Suffix nicht anzugeben.

### 5.2.2 sacego - ein Listenprogramm ablaufen lassen

Mit dem Programmaufruf *sacego* starten Sie ein ablauffähiges Listenprogramm. Dieses Programm wurde

- mit *saceprep* (siehe Abschnitt 5.2.1) oder
- im INFORMIX-Menüsystem (siehe Abschnitt 5.1) oder
- mit *isql* (siehe Kapitel 7.3.1)

übersetzt; sein Dateiname hat das Suffix *.arc*.

```
sacego [ -q ] [ -d datenbank ] datei [ parameterwerte ] ...
```

**-q**
unterdrückt alle Meldungen am Bildschirm, die keine Fehlermeldungen sind.

**-d** datenbank
legt eine andere Datenbank fest, für die die Liste erstellt werden soll. Die Liste muß natürlich zu dieser Datenbank passen.

Standardmäßig erstellt *sacego* die Liste für die Datenbank, die im Listen-Quellprogramm steht (siehe Abschnitt 5.3, DATABASE-Abschnitt).

datei
ist der Name der Datei, die das ablauffähige Listenprogramm enthält. Er muß das Suffix *.arc* haben, aber Sie brauchen das Suffix nicht anzugeben. Sie können auch die Namen von mehren Listenprogrammen angeben. Diese laufen dann nacheinander ab.

parameterwerte
Wenn für das Listenprogramm Parameter definiert sind (siehe Abschnitt 5.3, DEFINE-Abschnitt), dann muß man hier die Werte in der richtigen Reihenfolge hinschreiben. Zwischen zwei Werten steht immer ein Leerzeichen.

## 5.3 Aufbau eines Listen-Quellprogramms

Ein Listen-Quellprogramm besteht aus Abschnitten. In den einzelnen Abschnitten legen Sie fest:

- aus welcher Datenbank die Daten für die Liste kommen
- welche eigenen Variablen Sie im Listenprogramm verwenden
- ob während des Ablaufs des Listenprogramms Benutzereingaben erfolgen
- wie die äußere Form der Ausgabe ist, und welches Ausgabemedium Sie verwenden
- welche Daten aus welchen Tabellen Grundlage für die Liste sind
- wie Sie die Daten aufbereiten und ausgegeben

**Überblick über den Aufbau des Listenprogramms**

DATABASE-Abschnitt

[DEFINE-Abschnitt]

[INPUT-Abschnitt]

[OUTPUT-Abschnitt]

```
{SELECT-Abschnitt  }
{READ-Abschnitt    }
```

FORMAT-Abschnitt

Ein Listenquellprogramm besteht aus mindestens drei und maximal sechs Abschnitten. Die Reihenfolge der Abschnitte müssen Sie einhalten. Welche Bedeutung die Abschnitte haben, und wie ihre Syntax ist, steht auf den folgenden Seiten.

### Komentare im Listenprogramm

Sie dürfen an jeder Stelle eines Listen-Quellprogramms Kommentare schreiben, wo auch Leerzeichen stehen können. Zur Kennzeichnung eines Kommentars benutzen Sie wahlweise folgende Zeichen:

`#, --, { }`

Schreibweise:

`#` *kommentar* oder
`--` *kommentar* (ANSI-Schreibweise) oder
`{` *kommentar* `}`

### Hinweis

Im Anhang A.1 finden Sie ausführliche Beispiele für Listen-Quellprogramme.

## DATABASE-Abschnitt

Dieser Abschnitt muß der erste Abschnitt in Ihrem Quellprogramm sein. In ihm legen Sie fest, aus welcher Datenbank die Daten für die Liste kommen.

```
DATABASE

  {datenbank}
  {ASCII    }

END
```

datenbank
: Hier geben Sie den Namen der Datenbank an. Enthält das Programm den READ-Abschnitt, ist die Angabe einer gültigen Datenbank zwar Pflicht, die angegebene Datenbank wird jedoch nicht verwendet.

ASCII
: Schlüsselwort, das Sie anstelle des Datenbanknamens angeben können, wenn das Programm den READ-Abschnitt enthält.
Enthält der READ-Abschnitt die Angabe ORDER[ EXTERNAL] BY, so sollten Sie anstelle des Angabe ASCII eine gültige Datenbank angeben.

**Hinweis**

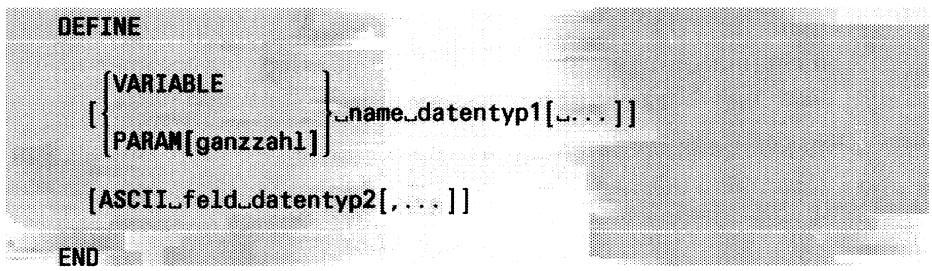
Sie können beim Ablauf auch die Daten einer anderen Datenbank verwenden, wenn Sie ein Listenprogramm von der Betriebssystemebene aus starten. (siehe Abschnitt 5.2.2).

## DEFINE-Abschnitt

Dieser Abschnitt ist wahlfrei für Listenprogramme, die den SELECT-Abschnitt verwenden. Listenprogramme, die statt SELECT den READ-Abschnitt verwenden, müssen auch den DEFINE-Abschnitt enthalten. Mit DEFINE definieren Sie für Listenprogramme mit

- SELECT-Abschnitt: Variablen und Parameter, die Sie in INPUT-, SELECT- oder FORMAT-Abschnitt verwenden.
- READ-Abschnitt: Namen und Datentypen für die einzelnen Felder der Eingabedatei. Außerdem Variablen und Parameter, die Sie in INPUT-, SELECT- oder FORMAT-Abschnitt verwenden.

Maximal 100 Variablen und Parameter können Sie im DEFINE-Abschnitt angeben.




VARIABLE
geben Sie an, wenn Sie für das Listenprogramm eine Variable definieren wollen. Man kann sie nur innerhalb des Programms mit einem Wert belegen.

PARAM
geben Sie an, wenn Sie für das Listenprogramm einen Parameter definieren wollen. Er ist eine Variable, die ihren Wert beim Aufruf des Listenprogramms auf der Betriebssystemebene oder aus dem Benutzermenü erhält. Der Benutzer kann das Listenprogramm nicht mehr im Menü LISTE starten, weil er dort keinen Parameter übergeben kann.

[ganzzahl]
Hier legen Sie mit *ganzzahl* die Positon des Parameters beim Aufruf auf der Betriebssystemebene fest. '1' steht für die erste Position nach dem Namen des Listenprogramms. Die eckigen Klammern [] müssen Sie eingeben.

name
Hier geben Sie der Variable einen Namen. Wenn Sie die Variable im SELECT-Abschnitt verwenden, dann müssen Sie dort dem Namen ein Dollarzeichen '$' voranstellen. Der Name darf nicht übereinstimmen mit dem Namen einer Tabellenspalte.

datentyp1
Hier legen Sie für die Variable einen Datentyp fest. Die Beschreibung der Datentypen finden Sie in Kapitel 4 des SQL-Handbuchs [1].

*Einschränkungen:*

- Der INFORMIX-ONLINE-Datentyp BYTE ist nicht erlaubt.
- Beim INFORMIX-ONLINE-Datentyp VARCHAR geben Sie nicht (max,min) an, sondern lediglich (max), wobei Sie mit *max* die maximale Länge festlegen, die berücksichtigt werden soll.
- Beim Datentyp MONEY dürfen Sie keine weiteren Angaben in Klammern machen.

ASCII
Hier definieren Sie für die im READ-Abschnitt verwendete Eingabedatei Feldnamen und Datentyp.

feld
Feldname. Die Anzahl der angegebenen Feldnamen muß mit der Anzahl der Felder in der Eingabedatei übereinstimmen.

datentyp2
ordnet dem Wert des angegebenen Feldes einen Datentyp zu. Die Beschreibung der Datentypen finden Sie in Kapitel 4 des SQL-Handbuchs [1].

*Einschränkungen:*

- Der INFORMIX-ONLINE-Datentyp BYTE ist nicht erlaubt.
- Beim INFORMIX-ONLINE-Datentyp VARCHAR geben Sie nicht (max,min) an, sondern lediglich (max), wobei Sie mit *max* die maximale Länge festlegen, die berücksichtigt werden soll.
- Beim Datentyp MONEY dürfen Sie keine weiteren Angaben in Klammern machen.

Wenn Sie hier einen Datentyp angeben, der nicht zum entsprechenden Wert in der Eingabedatei paßt, erkennt ACE den Fehler erst bei Ablauf des Listenprogramms.

**Beispiel**

Das folgende Beispiel zeigt die Definition einzelner Felder einer Eingabedatei, wie Sie im READ-Abschnitt verwendet wird.

```
DEFINE
   ASCII artikel_nr smallint, herstellercode char(3),
         bezeichnung char(15), preis money,
         liefereinheit char(4), stueckliste char(15)
END
```

## INPUT-Abschnitt

Dieser Abschnitt ist wahlfrei. Sie verwenden ihn, wenn der Benutzer nach Aufruf des Listenprogramms Werte eingeben soll. Das Programm wartet dann vor der Datenbereitstellung solange, bis der Benutzer die Werte eingegeben hat. Den Text der Eingabeaufforderung geben Sie vor. Maximal 40 Eingabeaufforderungen mit jeweils PROMPT FOR sind erlaubt.

```
INPUT
   {PROMPT␣FOR␣variable␣USING␣"text"}...
END
```

variable
: müssen Sie bereits im DEFINE-Abschnitt definiert haben. Für diese Variable fordert das Listenprogramm automatisch einen Eingabewert an. Wenn der Wert nicht zum Datentyp paßt, dann gibt ACE eine Fehlermeldung aus und die Liste wird nicht erstellt.

text
: erscheint als Eingabeaufforderung am Bildschirm. Er darf maximal 80 Zeichen lang sein und muß in Anführungsstrichen stehen.

### Hinweis

PROMPT FOR können Sie in folgenden Fällen nicht benutzen:

- Wenn Sie zum Ablaufzeitpunkt den Datenbanknamen eingeben wollen. Statt PROMPT FOR verwenden Sie hier den Schalter **-d** des ACE-Aufrufs *sacego* (siehe Abschnitt 5.2.2).
- Wenn Sie zum Ablaufzeitpunkt einen Dateinamen für die Ausgabe festlegen wollen. Dies können Sie nur im OUTPUT-Abschnitt angeben, eine Angabe zum Ablaufzeitpunkt ist nicht möglich.

**Beispiel**

```
DEFINE
  ...
  VARIABLE land char(2)          { Variable definieren          }
  ...
END

INPUT
  PROMPT FOR land USING          { Programm wartet bis Benutzer }
  "Bitte Bundesland angeben:"    { etwas eingibt.               }
END

...
```

## OUTPUT-Abschnitt

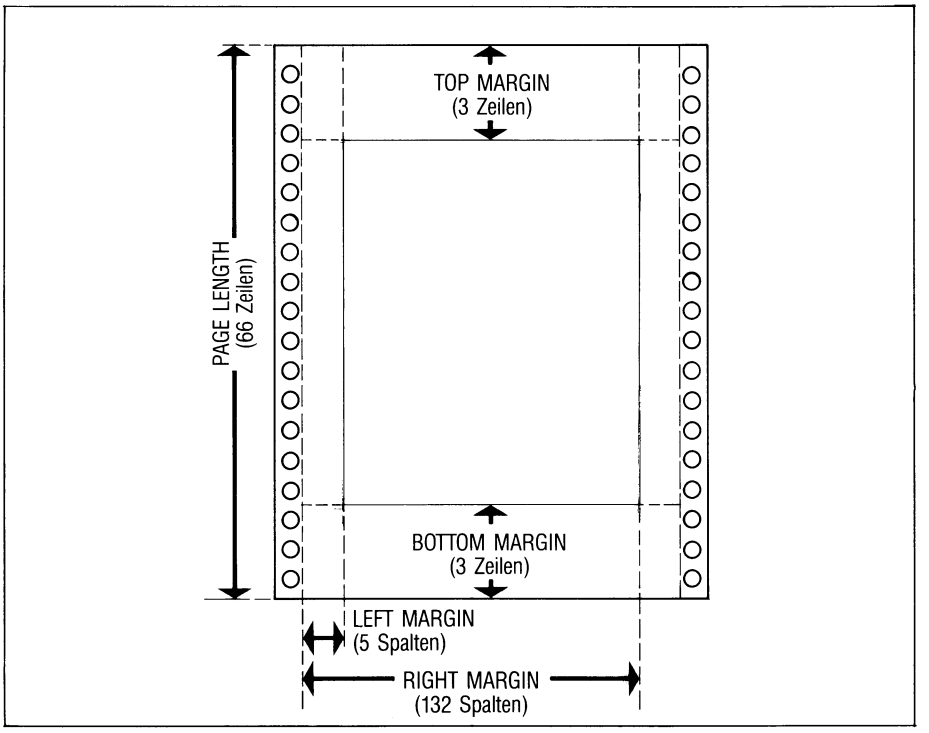
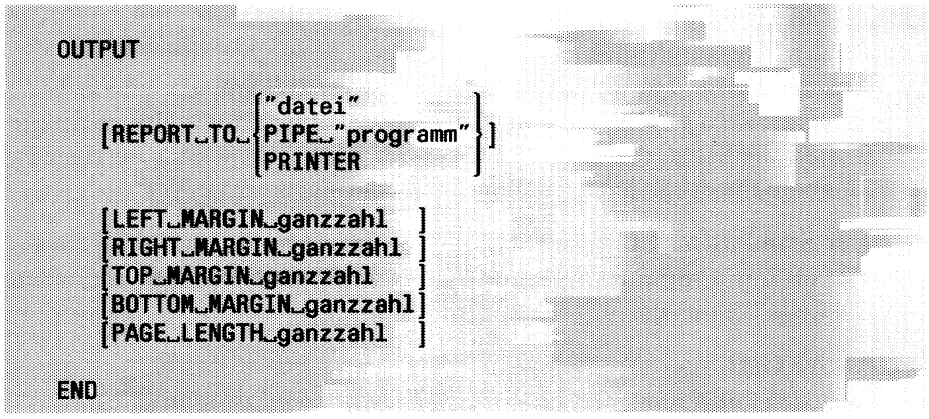



Bild 5-1 Listenränder und Seitenlängen

Dieser Abschnitt ist wahlfrei. In ihm legen Sie fest, ob die erstellte Liste in eine Datei geschrieben, über den Drucker ausgegeben oder an ein Programm weitergeleitet wird. Außerdem können Sie die Ränder der Liste und die Seitenlänge festlegen. Wenn ein Listen-Quellprogramm diesen Abschnitt nicht hat, dann gelten die Standardwerte, die bei den jeweiligen Operanden angegeben sind.




REPORT TO
definiert das Ausgabemedium.

Fehlt die Angabe REPORT, dann wird die Liste am Bildschirm ausgegeben. Im Standardlistenprogramm gilt: OUTPUT REPORT TO PRINTER.

datei
Name der Datei, in die die Liste geschrieben werden soll. Er muß in Anführungszeichen stehen.

PIPE "programm"
Name des Programms, an das die Liste weitergeleitet werden soll. Er muß in Anführungszeichen stehen.

PRINTER
leitet die Liste an das Druckprogramm weiter, das mit der Umgebungsvariablen DBPRINT vereinbart ist (siehe Kapitel 7.2).

Standardmäßig gilt als Druckprogramm das Betriebsystemkommando *lpr*, d.h. die Liste wird ausgedruckt.

LEFT MARGIN
Sie legen mit einem Wert für *ganzzahl* den linken Listenrand fest.

Der Standardwert für *ganzzahl* ist 5, d.h. 5 Spalten bleiben frei.

RIGHT MARGIN

Diesen Operanden brauchen Sie nur, wenn Sie im FORMAT-Abschnitt den Operanden EVERY ROW verwenden. Mit einem Wert für *ganzzahl* legen Sie die maximale Zeilenlänge fest, gezählt ab Spalte 0. Sind die Sätze, die der SELECT- oder READ-Abschnitt bereitstellt, länger, dann erhält jeder Spalteninhalt eine eigene Zeile.

Der Standardwert für *ganzzahl* ist 132. Beachten Sie, daß Ausgabegeräte oft nur 80 Spalten haben und einen Zeilenumbruch machen, wenn die Zeile länger ist.

TOP MARGIN

Sie legen mit einem Wert für *ganzzahl* den oberen Listenrand fest.

Der Standardwert für *ganzzahl* ist 3, d.h. die ersten 3 Zeilen bleiben frei.

BOTTOM MARGIN

Sie legen mit einem Wert für *ganzzahl* den unteren Listenrand fest.

Der Standardwert für *ganzzahl* ist 3, d.h. die unteren 3 Zeilen bleiben frei.

PAGE LENGTH

Sie legen mit einem Wert für *ganzzahl* die Seitenlänge fest. Der Wert muß größer sein als die Summe von oberem und unterem Listenrand und Seitenkopf- und Seitenende-Zeilen (siehe FORMAT-Abschnitt).

Der Standardwert für *ganzahl* ist 66, d.h. eine Seite hat 66 Zeilen. Beachten Sie, daß das SINIX-Kommando *lpr* 72 Zeilen als Standard hat.

**Beispiel**

```
...

OUTPUT                          { Die Liste wird an das more-}
   REPORT TO PIPE "more"        { Kommando weitergeleitet: }
END                             { Bildschirmweise Ausgabe.  }

...
```

## SELECT-Abschnitt

Jedes Listen-Quellprogramm muß einen SELECT- oder einen READ-Abschnitt haben.
SELECT stellt die Sätze aus einer Tabelle bereit, die in die Liste kommen. Die Datenbank haben Sie im DATABASE-Abschnitt definiert; wie die Sätze für die Liste aufbereitet werden, legen Sie im FORMAT-Abschnitt fest.

```
SELECT-anweisung[;SELECT-anweisung]...
END
```

Im SELECT-Abschnitt können mehrere SELECT-Anweisungen stehen. Nur die letzte SELECT-Anweisung stellt die Sätze für die Liste bereit. Die vorhergehenden Anweisungen liefern nur Zwischenergebnisse.

Die Syntax einer SELECT-Anweisung ist im SQL-Handbuch [1] beschrieben. Zusätzlich müssen Sie folgendes beachten:

- Die Tabellennamen müssen mit dem Eigentümernamen qualifiziert werden, wenn die betroffene Datenbank eine ANSI-Datenbank ist und das Listenprogramm nicht unter der Benutzerkennung abläuft, die Eigentümer der Datenbank ist.
- Bis auf die letzte SELECT-Anweisung müssen alle Anweisungen den Zusatz INTO TEMP enthalten. Die letzte darf den Zusatz nicht enthalten.
- Spalten des INFORMIX-ONLINE-Datentyps BYTE dürfen in SELECT nicht verwendet werden.
- Im Ergebnissatz der letzten SELECT-Anweisung darf kein Spaltenname doppelt vorkommen. Geben Sie daher einer Spalte einen neuen Namen, wenn zwei Tabellen den gleichen Spaltennamen verwenden.

- Nur die letzte SELECT-Anweisung darf ORDER BY haben. Sie dürfen hier bei den Spaltennamen keinen Tabellennamen angeben.
  Ggf. müssen Sie in *spaltenauswahl* einen mit *tabelle* qualifizierten Spaltennamen in einen einfachen Spaltennamen umbenennen und diesen Spaltennamen dann in ORDER BY verwenden (siehe auch Beispiel 2, *aufnr*).
  Sie dürfen in ORDER BY auch keine Spalte des Datentyps TEXT angeben.
- Wenn Sie in einer SELECT-Anweisung eine Variable verwenden, dann müssen Sie vor ihren Namen ein Dollarzeichen '$' schreiben. Variablen definieren Sie im DEFINE-Abschnitt.
- Innerhalb des SELECT-Abschnitts dürfen Sie keine Kommentare schreiben.

**Beispiel 1**

```
SELECT *
   FROM kunde
   ORDER BY nachname
END
```

**Beispiel 2** (aus dem Listenprogramm *auftrag1.ace*)

```
SELECT
   auftrag.auftrags_nr aufnr,
   auftragsdatum, kunden_nr,
   fremd_nr, lieferdatum, zustellgebuehr,
   zahldatum,

   posten.auftrags_nr, artikel_nr, herstellercode,
   menge, gesamtpreis

   FROM auftrag, posten

   WHERE auftrag.auftrags_nr = posten.auftrags_nr

   ORDER BY aufnr
END
```

**Beispiel 3** (aus dem Listenprogramm *kliste2.ace*)

```
SELECT
   kunden_nr,
   vorname,
   nachname,
   firma,
   ort,
   bundesland,
   plz,
   telefon

   FROM kunde

   WHERE bundesland MATCHES $diesland

   ORDER BY plz, nachname
END
```

## READ-Abschnitt

Jedes Listen-Quellprogramm muß einen SELECT- oder einen READ-Abschnitt haben.
READ liest Sätze aus einer ASCII-Eingabedatei. Das Format der Eingabesätze muß genau dem Format entsprechen, das die SQL-Anweisung UNLOAD erzeugt (siehe SQL-Handbuch [1]). Die Eingabedatei kann also eine von UNLOAD erzeugte Ausgabedatei sein, eine über die PERFORM-Funktion PRINT erzeugte Ausgabedatei im UNLOAD-Format oder eine beliebige Datei, deren Sätze im UNLOAD-Format enthalten sind.

Im READ-Abschnitt vereinbaren Sie den Namen der Eingabedatei, das Trennzeichen, das die einzelnen Felder des Eingabesatzes voneinander trennt und die Sortierung der Ausgabe.

Wenn Sie READ verwenden, so müssen Sie auch einen DEFINE-Abschnitt anlegen. Dort definieren Sie über die ASCII-Anweisung die einzelnen Felder der Eingabedatei und ordnen den Feldinhalten einen Datentyp zu.

Obwohl die Sätze der Eingabedatei keiner Datenbank zugeordnet sein müssen, muß im DATABASE-Abschnitt eine (beliebige) existierende Datenbank oder das Schlüsselwort ASCII angegeben werden.

Wenn Sie ORDER BY oder ORDER EXTERNAL BY verwenden, sollten Sie im DATABASE-Abschnitt anstelle des ASCII-Schlüsselwortes einen gültigen Datenbanknamen angeben.

```
READ
 "datei"[␣DELIMITER␣"zeichen"]
   [␣ORDER[␣EXTERNAL]␣BY␣feld[␣{ASC|DESC}][,...]
END
```

"datei"
Name der Eingabedatei. Ist die Datei nicht im aktuellen Dateiverzeichnis enthalten, müssen Sie den absoluten Pfadnamen angeben. Die Sätze der Datei dürfen keine Werte des Datentyps BYTE (INFORMIX-ONLINE) enthalten.

DELIMITER "zeichen"
Zeichen, das die einzelnen Felder des Eingabesatzes voneinander trennt. Fehlt die Angabe, so gilt das Trennzeichen, das in der Umgebungsvariablen DBDELIMITER vereinbart ist (siehe 7.2). Ist DBDELIMITER nicht gesetzt, so gilt das Trennzeichen "|".
*Beachten Sie:* Bei ACE sind folgende Trennzeichen verboten: Leerzeichen ␣, Anführungszeichen ".

ORDER BY
sortiert die Eingabesätze nach den angegebenen Feldern (siehe *feld*).

ORDER EXTERNAL BY
gibt an, daß die Eingabesätze bereits nach den angegebenen Feldern sortiert sind (siehe *feld*). Enthält der FORMAT-Abschnitt BEFORE GROUP- oder AFTER GROUP-Kontrollblöcke für zwei oder mehr Felder, dann muß über ORDER EXTERNAL BY die Sortierreihenfolge dieser Felder festgelegt werden.

feld
Name des Feldes, nach dem sortiert werden soll bzw., wenn EXTERNAL angegeben ist, Feld, nach dem die Datei sortiert ist. Den Feldnamen geben Sie entsprechend der Definition im DEFINE-Abschnitt an. Ein Feld vom Datentyp TEXT darf nicht angegben werden. Sie können mehrere Feldnamen angeben.

Die Reihenfolge der Angaben bezeichnet gleichzeitig die Reihenfolge der Sortierung: Erst werden alle Sätze in Reihenfolge des ersten angegebenen Feldes sortiert. Sind Sätze mit gleichem Wert in diesem Feld vorhanden, werden diese nach dem zweiten angegebenen Feld sortiert usw.

ASC
Standardwert: aufsteigende Sortierung.

DESC
absteigende Sortierung.

**Beispiel 1**

ACE soll Sätze aus der Eingabedatei *artikel.unl* lesen. Die Felder der Eingabesätze sind durch das Trennzeichen Doppelpunkt ":" voneinander getrennt. Die Ausgabe soll absteigend nach dem Feld *preis* sortiert werden. Sätze mit gleichem Preis sollen aufsteigend nach dem Feld *bezeichnung* sortiert werden.

```
READ
   "artikel.unl" DELIMITER ":"
   ORDER BY preis DESC, bezeichnung
END
```

**Beispiel 2**

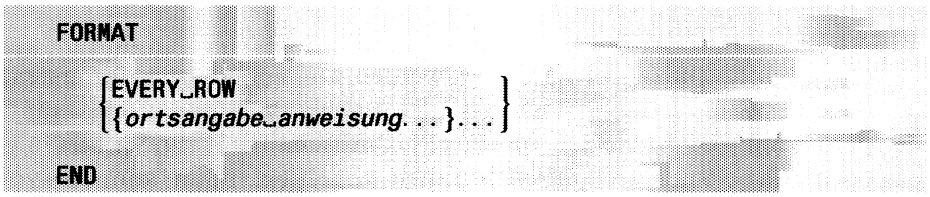
Im folgenden Beispiel enthält die Eingabedatei *artikel.unl* Sätze, die nach den Feldern *bezeichnung* und *preis* bereits sortiert sind. Die AFTER GROUP-Anweisungen im FORMAT-Abschnitt beziehen sich auf die Sortierreihenfolge dieser Felder in der Eingabedatei.

```
READ
   "artikel.unl" DELIMITER ":"
   ORDER EXTERNAL BY bezeichnung, preis
END

...

FORMAT
...
AFTER GROUP OF bezeichnung
...
AFTER GROUP OF preis
...
...
```

## FORMAT-Abschnitt

Diesen Abschnitt muß jedes Listen-Quellprogramm haben. In ihm legen Sie fest, wie die Sätze bearbeitet werden und wie die Daten in die Liste kommen. Die Sätze liefert die letzte bzw. einzige SELECT-Anweisung im SELECT-Abschnitt oder die Eingabedatei, die im READ-Abschnitt angegeben ist. Für das Aussehen Ihrer Liste wählen Sie

- das Standardlayout oder
- ein von Ihnen selbst entworfenes Layout. In diesem Fall besteht Ihr Entwurf aus:
  - Anweisungen, die festlegen, was das Listenprogramm tun soll und
  - Ortsangaben, die festlegen, an welcher Stelle der Liste bzw. bei welchem Verarbeitungsstand das Listenprogramm die Anweisung ausführen soll.




EVERY ROW

Standardlayout:
gibt die Sätze so aus, wie sie vom SELECT- oder READ-Abschnitt kommen. Sind die Sätze kürzer als 132 Zeichen, dann kommt in jede Zeile ein Satz; sind sie länger, dann kommt in jede Zeile nur ein Spalte (SELECT) bzw. nur ein Feld (READ). Die Satzlänge kann man mit RIGHT MARGIN im OUTPUT-Abschnitt ändern.

Ein Standardlistenprogramm enthält im FORMAT-Abschnitt die Angabe EVERY ROW.

ortsangabe anweisung

verwenden Sie, um selbst ein Layout zu entwerfen. Die Beschreibung dieser Angaben finden Sie in den nachfolgenden Abschnitten *Ortsangabe im FORMAT-Abschnitt* und *Anweisung im FORMAT-Abschnitt.*

## Ortsangabe im FORMAT-Abschnitt

Mit der Ortsangabe legen Sie fest, an welcher Stelle der Liste bzw. bei welchem Verarbeitungsstand das Listenprogramms die die Anweisung ausführen soll. Auf eine Ortsangabe folgen immer eine oder mehrere Anweisungen. (siehe nächster Abschnitt *Anweisungen*). Diese werden entsprechend der Ortsangabe ausgeführt.

```
                    ⎧FIRST␣PAGE␣HEADER
                    ⎪PAGE␣HEADER
                    ⎪BEFORE␣GROUP␣OF␣ordnungsbegriff
ortsangabe: :=      ⎨ON␣EVERY␣ROW
                    ⎪AFTER␣GROUP␣OF␣ordnungsbegriff
                    ⎪ON␣LAST␣ROW
                    ⎩PAGE␣TRAILER
```

FIRST PAGE HEADER

Die nachfolgenden Anweisungen werden nur am Anfang der ersten Seite ausgeführt. Dabei bleiben standardmäßig drei Zeilen frei (veränderbar mit TOP MARGIN im OUTPUT-Abschnitt). Wenn Sie hier mit einer Anweisung auf Tabellenspalten zugreifen, dann kann es zu unvorhersehbaren Reaktionen kommen.

PAGE HEADER

Die nachfolgenden Anweisungen werden an jedem Seitenanfang ausgeführt; jedoch nicht auf der ersten Seite, wenn Sie auch FIRST PAGE HEADER verwenden. Wenn Sie hier mit einer Anweisung auf Tabellenspalten zugreifen, dann kann es zu unvorhersehbaren Reaktionen kommen.

BEFORE GROUP OF ordnungsbegriff

*ordnungsbegriff* muß in derselben Schreibweise bei ORDER BY in der letzten SELECT- oder der READ-Anweisung vorkommen. Die nachfolgenden Anweisungen werden am Anfang der Liste und vor jedem Satz ausgeführt, bei dem sich der Inhalt von *ordnungsbegriff* oder der Inhalt eines übergeordneten Ordnungsbegriffs ändert.

Die Ortsangabe BEFORE GROUP OF kann sooft vorkommen, wie es Ordnungsbegriffe bei ORDER BY in der SELECT- oder READ-Anweisung gibt. Einen übergeordneter Ordnungsbegriff haben Sie dort an der vorhergehenden Stelle genannt.
Zu beachten: Hinweis am Ende des Abschnitts.

ON EVERY ROW
Die nachfolgenden Anweisungen werden bei jedem Satz ausgeführt.
Zu beachten: Hinweis am Ende des Abschnitts.

AFTER GROUP OF ordnungsbegriff
*ordnungsbegriff* muß in derselben Schreibweise bei ORDER BY in der letzten SELECT- oder in der READ-Anweisung vorkommen. Die nachfolgenden Anweisungen werden nach jedem Satz ausgeführt, bei dem sich der Inhalt von *ordnungsbegriff* oder der Inhalt eines übergeordneten Ordnungsbegriffs ändert und am Ende der Liste.
Die Ortsangabe AFTER GROUP OF kann sooft vorkommen, wie es Ordnungsbegriffe bei ORDER BY in der SELECT- oder READ-Anweisung gibt. Einen übergeordneter Ordnungsbegriff haben Sie dort an der vorhergehenden Stelle genannt.
AFTER GROUP OF wird vor ON LAST ROW ausgeführt. Zwischen zwei Gruppen wird AFTER GROUP OF vor BEFORE GROUP OF ausgeführt.
Zu beachten: Hinweis am Ende des Abschnitts.

ON LAST ROW
Die nachfolgenden Anweisungen werden nach dem letzten Satz ausgeführt.

PAGE TRAILER
Die nachfolgenden Anweisungen werden an jedem Seitenende ausgeführt.

**Hinweis:** zu den Ortsangaben BEFORE GROUP OF, AFTER GROUP OF, ON EVERY ROW

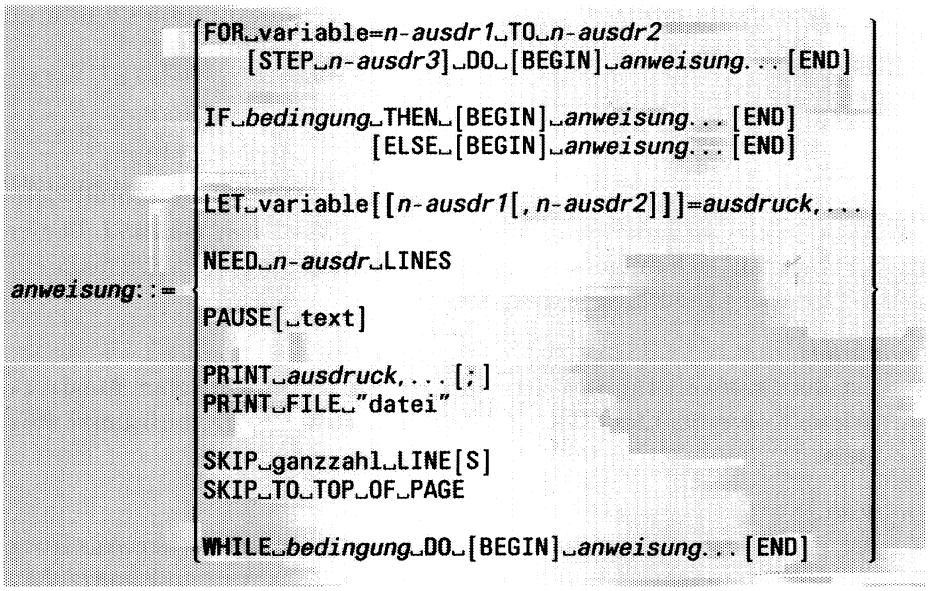
Wenn Sie diese Ortsangaben zusammen in einem Listenprogramm verwenden, so bearbeitet ACE diese Angaben in folgender Reihenfolge:

```
before group of a
       before group of b
              before group of c
                     on every row
              after group of c
       after group of b
after group of a
```

Die Angaben a, b und c seien Spalten, die in der ORDER ...BY-Angabe des SELECT- oder READ-Abschnitts angegeben sind.

## Anweisung im FORMAT-Abschnitt

Nach jeder Ortsangabe im FORMAT-Abschnitt können eine oder mehrere Anweisungen stehen.




Welche numerischen Ausdrücke, Ausdrücke und Bedingungen für Listenprogramme möglich sind, finden Sie in diesem Abschnitt bei den Kolumnentiteln FORMAT/Ausdruck bzw. FORMAT/Bedingung beschrieben.

## FOR-Anweisung

Steht am Anfang einer FOR-Schleife. Diese wird solange durchlaufen, bis *variable* den Wert von *n-ausdr2* annimmt.

```
FOR␣variable=n-ausdr1␣TO␣n-ausdr2[␣STEP␣n-ausdr3]
   DO␣[BEGIN]
   anweisung...
   [END]
```

variable
: ist die Schleifenvariable. Sie müssen sie im DEFINE-Abschnitt definiert haben. *variable* erhält am Anfang den Wert von *n-ausdr1*. Dieser Wert erhöht sich mit jedem Schleifendurchlauf um eins, wenn nicht mit STEP etwas anderes vereinbart ist.

*n-ausdr1*
: ist der Startwert für *variable*. *n-ausdr1* ist ein numerischer Ausdruck (siehe Kolumnentitel FORMAT/Ausdruck); sein Wert muß ganzzahlig sein.

*n-ausdr2*
: ist der Endwert für *variable*. *n-ausdr2* ist ein numerischer Ausdruck (siehe Kolumnentitel FORMAT/Ausdruck); sein Wert muß ganzzahlig sein.

STEP *n-ausdr3*
: ist die Schrittweite für die Erhöhung von *variable*. *n-ausdr3* ist ein numerischer Ausdruck (siehe Kolumnentitel FORMAT/Ausdruck); sein Wert muß positiv und ganzzahlig sein.

  Fehlt die Angabe STEP, dann ist die Standard-Schrittweite = 1.

BEGIN ... END
: geben Sie an, wenn in der FOR-Schleife mehr als eine Anweisung steht.

*anweisung*
: Hier kann jede beliebige Anweisung stehen, die man im FORMAT-Abschnitt verwenden kann. Mehrere Anweisungen müssen Sie mit BEGIN und END einschließen.

## IF-Anweisung

Diese Anweisung entscheidet, ob Anweisungen ausgeführt werden. Ist die Bedingung erfüllt, dann werden die Anweisungen des THEN-Zweigs ausgeführt. Ist die Bedingung nicht erfüllt, dann werden die Anweisungen des THEN-Zweigs nicht ausgeführt. Hat die IF-Anweisung einen ELSE-Zweig, dann werden die Anweisungen des ELSE-Zweigs ausgeführt.

```
IF␣bedingung␣THEN␣[BEGIN]␣anweisung... [END]
                [ELSE␣[BEGIN]␣anweisung... [END]]
```

*bedingung*
muß erfüllt sein, damit die Anweisungen des THEN-Zweigs ausgeführt werden. Sonst wird er ELSE-Zweig ausgeführt, falls er vorhanden ist. Für den Aufbau einer Bedingung siehe Kolumnentitel FORMAT/ Bedingung.

BEGIN ... END
geben Sie an, wenn im THEN- bzw. im ELSE-Zweig mehr als eine Anweisung steht.

*anweisung*
Hier kann jede beliebige Anweisung stehen, die man im FORMAT-Abschnitt verwenden kann. Mehrere Anweisungen müssen Sie mit BEGIN und END einschließen. Man kann bis zu 128 IF-Anweisungen ineinander verschachteln.

Wenn Sie IF bei den Ortsangben FIRST PAGE HEADER, PAGE HEADER oder PAGE TRAILER verwenden, müssen die Anweisungen im THEN- und ELSE-Zweig die gleiche Anzahl von Zeilen drukken, da ACE den Platz dafür im voraus reserviert.

**Beispiel** (aus dem Listenprogramm *post3.ace*)

```
...
IF i = sp_anz THEN
BEGIN
   PRINT z_kette1 CLIPPED
   PRINT z_kette2 CLIPPED
   PRINT z_kette3 CLIPPED
   SKIP 1 line
   LET z_kette1 = " "
   LET z_kette2 = " "
   LET z kette3 = " "
   LET i  = 1
END
ELSE
   LET i = i + 1
...
```

## LET-Anweisung

Mit dieser Anweisung weisen Sie einer Variablen einen Wert zu.

```
LET variable[[n-ausdr1[,n-ausdr2]]]=ausdruck,...
```

variable
Die Variable müssen Sie im DEFINE-Abschnitt definiert haben.

Wenn sie einer INTEGER- oder SMALLINT-Variablen einen Wert mit Nachkommastellen zuweisen, so wird nur der ganzzahlige Teil des Wertes berücksichtigt.

[*n-ausdr1,n-ausdr2*]
dürfen Sie nur verwenden, wenn *variable* vom Typ CHAR ist. *n-ausdr1* und *n-ausdr2* müssen ganze Zahlen ergeben und definieren einen Ausschnitt aus der Variablen. Die Zahlen geben die erste und letzte Position des Ausschnitts an. Wenn Sie *n-ausdr2* weglassen, beginn der Ausschnitt an der Postition von *n-ausdr1* und geht bis zum Ende der Variablen. Die eckigen Klammern [] müssen Sie angeben.

*ausdruck*
Der Wert von *ausdruck* muß mit dem Typ der Variablen verträglich sein. Für den Aufbau eines Ausdrucks siehe Kolumnentitel FORMAT/ Ausdruck. Eine alphanumerische Konstante oder ein Datum muß in Anführungszeichen eingeschlossen sein. Kommazahlen müssen den Dezimalpunkt enthalten, wenn Kommastellen existieren. Die Form für Datumangaben hängt von der Umgebungsvariablen DBDATE ab (siehe Kapitel 7.2). *ausdruck* dürfen Sie nur mehrmals angeben, wenn die Variable vom Typ CHAR ist. Sie erhält dann den Wert, der sich durch Aneinanderfügen der Einzelwerte ergibt.

**Beispiel** (aus dem Listenprogramm *post3.ace*)

```
...
DEFINE
   ...
   VARIABLE a_breite SMALLINT
   VARIABLE platz    SMALLINT
   VARIABLE sp_anz   SMALLINT
   VARIABLE i        SMALLINT
END
...
FORMAT
   ...
   LET i = 1
   LET a_breite = 72/sp_anz
   LET platz = 8/sp_anz
   ...
END
```

## NEED-Anweisung

Diese Anweisung bewirkt einen Seitenvorschub, wenn auf der aktuelle Seite der Liste nicht mehr genügend Zeilen frei sind.

```
NEED␣n-ausdr␣LINES
```

*n-ausdr*
muß eine Zahl ergeben. NEED sorgt dafür, daß sich die angegebene Anzahl von Zeilen auf einer Seite befindet und nicht durch einen Seitenwechsel getrennt wird.

## PAUSE-Anweisung

Diese Anweisung bewirkt, daß die Ausgabe der Liste am Bildschirm angehalten wird, bis der Benutzer die Taste [↵] drückt. Die Anweisung bewirkt nichts, wenn die Liste nicht am Bildschirm ausgegebene wird.

```
PAUSE[␣"text"]
```

text
geben Sie als alphanumerische Konstante an. PAUSE gibt dann diesen Text aus. Die Anführungszeichen müssen Sie schreiben.

## PRINT-Anweisung

Diese Anweisung gibt eine Zeile der Liste aus und endet standardmäßig mit einem Zeilenvorschub. Ist WORDWRAP angegeben, ist die Ausgabe entsprechend mehrzeilig (siehe WORDWRAP unter Kolumnentitel FORMAT/ausdruck).

```
PRINT␣ausdruck,...[;]
```

*ausdruck*
gibt die PRINT-Anweisung aus. Für den Aufbau eines Ausdrucks siehe Kolumnentitel FORMAT/Ausdruck.
Beim Datentyp TEXT wirkt die PRINT-Anweisung entsprechend der PRINT FILE-Anweisung für eine Datei.
Wenn Sie das Ausgabeformat von Spalten nicht durch CLIPPED oder USING bestimmen, gibt PRINT folgende Standardlängen aus:

| Spaltentyp | Standardlänge |
|---|---|
| CHAR | entsprechend Spaltenlänge |
| SMALLINT | 6 mit Vorzeichen |
| INTEGER | 11 mit Vorzeichen |
| DECIMAL | entsprechend Stellenanzahl der Spalte, zuzüglich Vorzeichen und Komma |
| SMALLFLOAT | 14 mit Vorzeichen und Komma |
| FLOAT | 14 mit Vorzeichen und Komma |
| MONEY | entsprechend Stellenanzahl des Spalte, zuzüglich Vorzeichen und Komma |
| DATE | 10 |
| SERIAL | 11 |
| DATETIME | entsprechend den angegebenen Komponenten |
| INTERVAL | entsprechend den angegebenen Komponenten |
| VARCHAR | entsprechend der Länge des Inhalts |
| TEXT | entsprechend der Länge des Inhalts |

; Mit dem Semikolon ; am Ende unterdrücken Sie den Zeilenvorschub.

**Beispiel**

```
...
FIRST PAGE HEADER
PRINT COLUMN 33, "KUNDENLISTE"
PRINT COLUMN 33, "-----------"
SKIP 2 LINES
PRINT "Liste fuer Bundesland ", diesland
SKIP 2 LINES
PRINT "NUMMER",
   COLUMN 9, "NAME",
   COLUMN 32, "PLZ",
   COLUMN 40, "ORT",
   COLUMN 62, "TELEFON"
SKIP 1 LINE

PAGE HEADER
PRINT "NUMMER",
   COLUMN 9, "NAME",
   COLUMN 32, "PLZ",
   COLUMN 40, "ORT",
   COLUMN 62, "TELEFON"
SKIP 1 LINE

ON EVERY ROW
PRINT kunden_nr USING "####",
   COLUMN 9, vorname CLIPPED, 1 SPACE, nachname CLIPPED,
   COLUMN 32, plz,
   COLUMN 40, ort CLIPPED, ", " , bundesland,
   COLUMN 62, telefon
...
```

## PRINT FILE-Anweisung

Diese Anweisung gibt den Inhalt einer Textdatei in der Liste aus. Sie können so beispielsweise Standardbriefe erstellen.

```
PRINT␣FILE␣"datei"
```

datei
Hier geben Sie den Namen der Datei bzw. den Pfadnamen der Datei an, deren Inhalt in die Liste soll.

## SKIP-Anweisung

Diese Anweisung erzeugt in der Liste Leerzeilen oder bewirkt einen Seitenvorschub.

```
SKIP␣{ganzzahl␣LINE[S]}
     {TO␣TOP␣OF␣PAGE  }
```

ganzzahl
Hier geben Sie eine positive ganze Zahl an. Entsprechend viele Leerzeilen kommen dann in die Liste. Das *S* bei LINE können Sie schreiben oder weglassen.

TO TOP OF PAGE
bewirkt immer einen Seitenvorschub. Sie dürfen TO TOP OF PAGE nicht verwenden bei den Ortsangaben FIRST PAGE HEADER, PAGE HEADER und PAGE TRAILER.

## WHILE-Anweisung

Steht am Anfang einer WHILE-Schleife. Diese wird solange durchlaufen, wie die Schleifenbedingung erfüllt ist.

```
WHILE bedingung
  DO [BEGIN]
  anweisung...
  [END]
```

*bedingung*
ist die Schleifenbedingung. Die Schleife wird solange durchlaufen, wie die Bedingung erfüllt ist. Für den Aufbau einer Bedingung siehe Kolumnentitel FORMAT/Bedingung.

BEGIN ... END
geben Sie an, wenn in der WHILE-Schleife mehr als eine Anweisung steht.

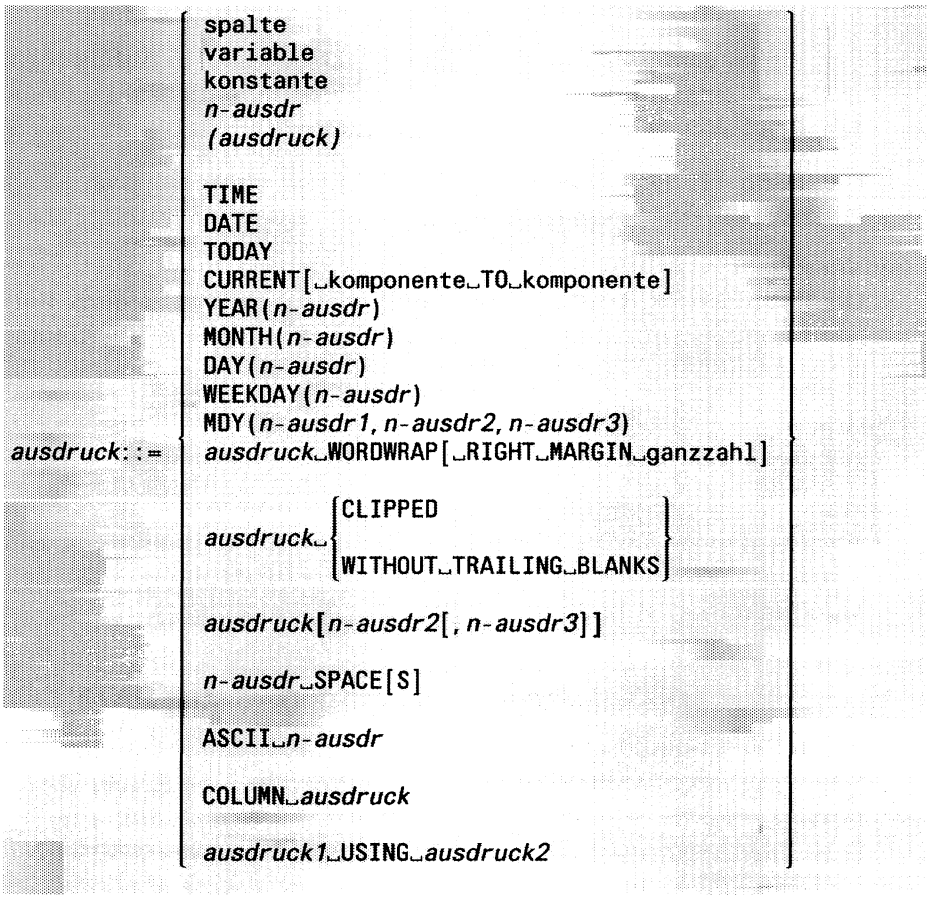
*anweisung*
Hier kann jede beliebige Anweisung stehen, die man im FORMAT-Abschnitt verwenden kann. Mehrere Anweisungen müssen Sie mit BEGIN und END einschließen.

## Operatoren bei Ausdrücken und Bedingungen

In Ausdrücken und bei Bedingungen kommen Operatoren vor. Die folgende Tabelle zeigt, wie deren Rangfolge bei der Auswertung eines Ausdrucks oder einer Bedingung ist. Zuerst wird ein Operator mit Rangfolge 1, dann mit Rangfolge 2 usw. ausgewertet. Will man eine andere Reihenfolge bei der Auswertung, dann muß man entsprechend Klammern ( ) setzen.

| Operator | Wirkung | Rangfolge |
|---|---|---|
| – | negativer Wert | 1 |
| ** | vor Exponenten | 2 |
| * | Multiplikation | 3 |
| / | Division | 3 |
| + | Addition | 4 |
| – | Subtraktion | 4 |
| IS [NOT] NULL | Abfrage NULL-Wert | 5 |
| MATCHES | Passende Zeichenfolge | 5 |
| = | ist gleich | 5 |
| != | ungleich | 5 |
| <> | ungleich | 5 |
| >= | größer oder gleich | 5 |
| <= | kleiner oder gleich | 5 |
| NOT | logisch *nicht* | 6 |
| AND | logisch *und* | 7 |
| OR | logisch *oder* | 8 |

## Ausdrücke für die Anweisungen




*n-ausdr* steht für numerische Ausdrücke. Diese Ausdrücke sind nach der Beschreibung der anderen Ausdrücke beschrieben.

spalte
: muß zu den Sätzen gehören, die die letzte SELECT-Anweisung im SELECT-Abschnitt oder der READ-Abschnitt liefert.

variable
muß im DEFINE-Abschnitt mit VARIABLE oder PARAM definiert sein.

konstante
Eine alphanumerische Konstante oder ein Datum muß in Anführungszeichen " eingeschlossen sein. Kommazahlen müssen den Dezimalpunkt enthalten, wenn Kommastellen existieren.

Die Form für Datumangaben hängt von der Umgebungsvariablen DBDATE ab (siehe Kapitel 7.2).

*(ausdruck)*
Die Klammern brauchen Sie, um Teile von Ausdrücken zu einer Einheit zusammenzufassen (Rangfolge der Operatoren siehe bei Kolumnentitel FORMAT/Operatoren).

TIME
gibt die aktuelle Uhrzeit als alphanumerische Konstante aus.

DATE
gibt das aktuelle Tagesdatum als alphanumerische Konstante aus. Die Konstante enthält ein Kürzel für den Tagesnamen und eins für den Monatsnamen (z.B. `Mi 23. Nov. 1987`).

TODAY
liefert das aktuelle Tagesdatum in der Form des Datentyps DATE.

CURRENT [komponente TO komponente]
liefert das aktuelle Tagesdatum und die Uhrzeit. Eine Beschreibung dieser Zeitfunktion finden Sie in Kapitel 5 des SQL-Handbuchs [1]. Fehlt die Angabe *komponente*, gilt als Standardwert: CURRENT YEAR TO FRACTION.

DATE(*n-ausdr*)
*n-ausdr* muß eine Zahl ergeben. DATE interpretiert die Zahl als die Nummer, die ein Tag bekommt, wenn man alle Tage ab dem 1.1.1900 durchnumeriert. DATE liefert das Datum dieses Tages in der Form des Datentyps DATE.

YEAR(*n-ausdr*)
berechnet wie DATE(...) oder DATETIME das Datum, liefert jedoch nur die vierstellige Jahreszahl.

MONTH(*n-ausdr*)
berechnet wie DATE(...) das Datum, liefert jedoch nur eine Zahl zwischen 1 und 12 für den Monat.

DAY(*n-ausdr*)
berechnet wie DATE(...) das Datum, liefert jedoch nur eine Zahl zwischen 1 und 31 für den Tag.

WEEKDAY(*n-ausdr*)
berechnet wie DATE(...) das Datum, liefert jedoch nur eine Zahl zwischen 0 und 6 für den Wochentag (0 entspricht Sonntag).

MDY(*n-ausdr1, n-ausdr2, n-ausdr3*)
konvertiert ein Datum in den Typ DATE, wobei gilt:

- *n-ausdr1* muß die Monatsangabe des Datums, also eine Zahl zwischen 1 und 12 ergeben
- *n-ausdr2* muß die Tagesangabe des Datums, also eine Zahl zwischen 1 und 28 ergeben oder muß abhängig vom Monat und Jahr 29, 30 oder 31 ergeben
- *n-ausdr3* muß die Jahresangabe des Datums ergeben

*ausdruck* WORDWRAP
*ausdruck* muß eine Zeichenkette sein (Datentypen CHAR, VARCHAR, oder TEXT). Gibt den Spalteninhalt mehrzeilig aus. Die Zeilenlänge ist maximal *ganzzahl*, wobei der Zeilenumbruch, wenn möglich bei ganzen Wörtern stattfindet. Enthält der Spaltenwert das ASCII-Zeichen 10 oder 13 (siehe ASCII-Tabelle im Anhang A.3), so findet ebenfalls ein Zeilenumbruch statt.

RIGHT MARGIN ganzzahl
Maximale Zeilenlänge. Diese Angabe gilt temporär anstelle der Vereinbarung RIGHT MARGIN im OUTPUT-Abschnitt.

*ausdruck* CLIPPED
*ausdruck* muß eine Zeichenkette sein (Datentypen CHAR, VARCHAR oder TEXT). Leerzeichen am Ende der Zeichenkette werden abgeschnitten.

*ausdruck* WITHOUT TRAILING BLANKS
wirkt genauso wie CLIPPED.

*ausdruck[n-ausdr1,n-ausdr2]*
definiert einen Ausschnitt aus der Zeichenkette *ausdruck*. *n-ausdr1* bezeichnet die Anfangsposition des Ausschnitts in der Zeichenkette und *n-ausdr2* die Endposition. Die beiden numerischen Ausdrücke müssen Zahlen ergeben. Wenn Sie *n-ausdr* nicht angeben ist das Ende

der Zeichenkette die Endposition des Ausschnitt. Die eckigen Klammern [] müssen Sie angeben.

*n-ausdr* SPACE[S]
liefert soviele Leerzeichen, wie der Wert von *n-ausdr* ist. Das *S* können Sie schreiben oder weglassen.

ASCII *n-ausdr*
liefert das Zeichen der ASCII-Tabelle, das dem Wert von *n-ausdr* entspricht (siehe ASCII-Tabelle im Anhang A.3).

COLUMN *n-ausdr*
ist nur in der PRINT-Anweisung sinnvoll. Die Ausgabeposition wird auf die Spalte verschoben, die dem Wert von *n-ausdr* entspricht. In den entstandenen Zwischenraum schreibt die PRINT-Anweisung Leerzeichen. Ist der Wert von *n-ausdr* kleiner als die aktuelle Schreibspalte, dann bleibt COLUMN ohne Wirkung.

*ausdruck1* USING *ausdruck2*
liefert eine Zeichenkette in der *ausdruck1* aufbereitet ist. Dabei gilt für den Inhalt:

- *ausdruck1* ist eine Zahl oder ein Datum
- *ausdruck2* ist eine Folge von Formatierzeichen. Wenn diese Folge eine alphanumerische Konstante ist, dann muß sie in Anführungszeichen eingeschlossen sein.

Wie *ausdruck2* aufgebaut wird, steht auf den folgenden Seiten.

**Datumsausgaben bei USING**

können mit folgenden Zeichen formatiert werden:

| ausdruck2 enthält | Datum enthält an derselben Stelle |
|---|---|
| dd | zweistellige Tageszahl |
| ddd | dreibuchstabiges Tageskürzel |
| mm | zweistellige Monatszahl |
| mmm | dreibuchstabiges Monatskürzel |
| yy | zweistellige Jahreszahl |
| yyyy | vierstellige Jahreszahl |
| ASCII-Zeichen | unverändert das ASCII-Zeichen |

**Numerische Werte bei USING**

können mit diesen Zeichen formatiert werden:

```
* & # < , . - + ( ) $
```

Das formatierte Ergebnis kann höchstens so viele Zeichen haben, wie die Formatierzeichenfolge. Die Bedeutung der Formatierzeichen entnehmen Sie den folgenden Tabellen.

| ausdruck2 enthält | Ergebnis enthält genau ein |
|---|---|
| $ | $ |
| ( | ( bei negativen Zahlen |
| - | - bei negativen Zahlen |
| + | + bei positiven Zahlen |
| | - bei negativen Zahlen |
| ) | ) an jeder Stelle, an der das Zeichen im Format angegeben ist, jedoch nur bei negativen Zahlen |
| . | , (Komma) und die Zahl wird dezimalstellen gerecht ausgerichtet. Ist bei DBMONEY *Punkt* gesetzt, dann steht Punkt statt Komma (DBMONEY siehe Kapitel 7.2). |

. (Punkt) an jeder Stelle, an der das Komma im Format angegeben ist, jedoch nur, wenn links vom Komma eine Zahl ungleich 0 steht. Ist bei DBMONEY *Punkt* gesetzt, dann steht Komma statt Punkt (DBMONEY siehe Kapitel 7.2). Sonst wird die Position besetzt wie die umgebenden Positionen.

Die Zeichen ( - ) werden bei positiven Zahlen durch ein Leerzeichen ersetzt. Jedes der Zeichen $ ( ) - + können Sie mehrfach hintereinander angeben. Damit erreichen Sie, daß das Zeichen mit der Größe der Zahl gleitet: Das Zeichen wird von ACE möglichst dicht links neben der Zahl plaziert. Dabei ist seine Position nach rechts hin jedoch beschränkt durch die letzte Position, die das Zeichen in der Formatierzeichenfolge einnimmt.

Die übrigen Zeichen sind reine Füllzeichen. Diese markieren Positionen, die eine Ziffer aufnehmen können. Sofern die Zahl nicht alle zur Verfügung stehenden Positionen besetzt, bekommt die Position eines der folgenden Standardzeichen:

| **ausdruck2 enthält** | **Im Ergebnis nimmt die entsprechende Position eine Ziffer auf oder sie** |
|---|---|
| # | wird mit einem Leerzeichen besetzt |
| & | wird mit 0 besetzt |
| * | wird mit * besetzt |
| < | entfällt ganz. Damit erreichen Sie eine linksbündige Ausgabe. |

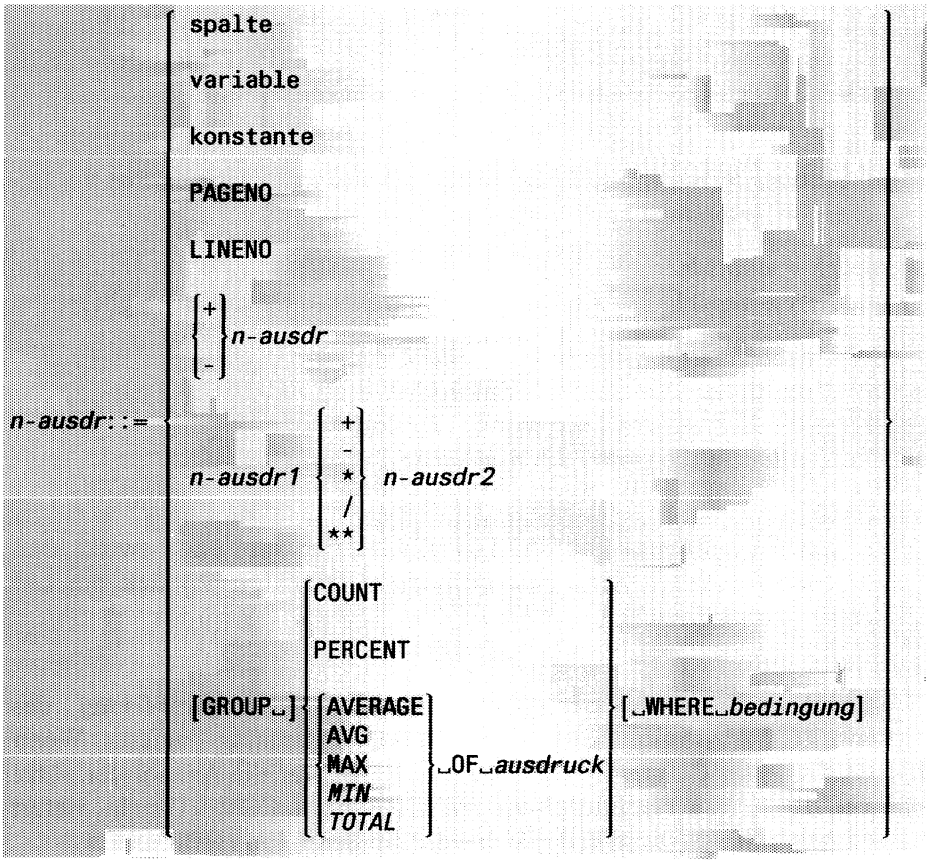
Wenn eine Zahl für das vorgegebene Format zu groß ist, belegt ACE das Ergebnis mit '*', um den Überlauf zu kennzeichnen.

## Beispiele für Formatierzeichenfolgen bei USING

| Formatierzeichenfolge | Wert | Formatierter Wert |
|---|---|---|
| "#####" | 0 | ␣␣␣␣␣ |
| "&&&&&" | 0 | 00000 |
| "$$$$$" | 0 | ␣␣␣␣$ |
| "*****" | 0 | ***** |
| "<<<<<" | 0 | ␣␣␣␣␣ NULL-Wert |
| "<<<, <<<" | 12345 | 12, 345 |
| "<<<, <<<" | 1234 | 1, 234 |
| "<<<, <<<" | 123 | 123 |
| "<<<, <<<" | 12 | 12 |
| "##, ###" | 12345 | 12. 345 |
| "##, ###" | 1234 | ␣1. 234 |
| "##, ###" | 123 | ␣␣␣123 |
| "##, ###" | 12 | ␣␣␣␣12 |
| "##, ###" | 1 | ␣␣␣␣␣1 |
| "##, ###" | -1 | ␣␣␣␣␣1 |
| "##, ###" | 0 | ␣␣␣␣␣0 |
| "&&, &&&" | 12345 | 12. 345 |
| "&&, &&&" | 1234 | 01. 234 |
| "&&, &&&" | 123 | 000123 |
| "&&, &&&" | 12 | 000012 |
| "&&, &&&" | 1 | 000001 |
| "&&, &&&" | 0 | 000000 |
| "$$, $$$" | 12345 | ****** Überlauf |
| "$$, $$$" | 1234 | $1. 234 |
| "$$, $$$" | 123 | ␣␣$123 |
| "$$, $$$" | 12 | ␣␣␣$12 |
| "$$, $$$" | 1 | ␣␣␣␣$1 |
| "$$, $$$" | 0 | ␣␣␣␣␣$ |
| "**, ***" | 12345 | 12. 345 |
| "**, ***" | 1234 | *1. 234 |
| "**, ***" | 123 | ***123 |
| "**, ***" | 12 | ****12 |
| "**, ***" | 1 | *****1 |
| "**, ***" | 0 | ****** |
| "##, ###. ##" | 12345. 67 | 12. 345, 67 |
| "##, ###. ##" | 1234. 56 | ␣1. 234, 56 |
| "##, ###. ##" | 123. 45 | ␣␣␣123, 45 |
| "##, ###. ##" | 12. 34 | ␣␣␣␣12, 34 |

| | | |
|---|---|---|
| "##,###.##" | 1.23 | ␣␣␣␣␣1,23 |
| "##,###.##" | 0.12 | ␣␣␣␣␣␣,12 |
| "##,###.##" | 0.01 | ␣␣␣␣␣␣,01 |
| "##,###.##" | -0.01 | ␣␣␣␣␣␣,01 |
| "##,###.##" | 0.00 | ␣␣␣␣␣␣,00 |
| | | |
| "&&,&&&.&&" | 12345.67 | 12.345,67 |
| "&&,&&&.&&" | 1234.56 | 01.234,56 |
| "&&,&&&.&&" | 123.45 | 000123,45 |
| "&&,&&&.&&" | 0.01 | 000000,01 |
| | | |
| "$$,$$$.$$" | 12345.67 | ********* Überlauf |
| "$$,$$$.$$" | 1234.56 | $1.234,56 |
| "$$,$$$.##" | 0.00 | $,00 |
| "$$,$$$.##" | 1234.56 | $1.234,56 |
| "$$,$$$.&&" | 0.00 | $,00 |
| "$$,$$$.&&" | 1234.00 | $1.234,00 |
| | | |
| "-##,###.##" | -12345.67 | -12.345,67 |
| "-##,###.##" | -1234.56 | - 1,234,56 |
| "-##,###.##" | -123.45 | - 123,45 |
| "-##,###.##" | -12.34 | - 12,34 |
| "--#,###.##" | -12.34 | - 12,34 |
| "---,###.##" | -12.34 | - 12,34 |
| "---,-##.##" | -12.34 | -12,34 |
| "---,--#.##" | -1.00 | -1,00 |
| | | |
| "-##,###.##" | 12345.67 | 12.345,67 |
| "-##,###.##" | 1234.56 | 1,234,56 |
| "-##,###.##" | 123.45 | 123,45 |
| "-##,###.##" | 12.34 | 12,34 |
| "--#,###.##" | 12.34 | 12,34 |
| "---,###.##" | 12.34 | 12,34 |
| "---,-##.##" | 12.34 | 12,34 |
| "---,---.##" | 1.00 | 1,00 |
| "---,---.--" | -.01 | -0,01 |
| | | |
| "----,---.&&" | -.01 | -0,01 |
| "-$$$,$$$.&&" | -12345.67 | -$12.345,67 |
| "-$$$.$$$.&&" | -1234.56 | - $1.234,56 |
| "-$$$,$$$.&&" | -123.45 | - $123,45 |
| "--$$,$$$.&&" | -12345.67 | -$12.345,67 |
| "--$$.$$$.&&" | -1234.56 | -$1.234,56 |
| "--$$,$$$.&&" | -123.45 | - $123,45 |
| "--$$.$$$.&&" | -12.34 | - $12,34 |
| "--$$,$$$.&&" | -1.23 | - $1,23 |

## Numerische Ausdrücke




spalte
: muß zu den Sätzen gehören, die die letzte SELECT-Anweisung im SELECT-Abschnitt oder die READ-Anweisung liefert. Eine Spalte des Datentyps TEXT ist nicht erlaubt. Bei den Datentypen CHAR und VARCHAR muß der Wert allerdings eine Zahl sein.

variable
: muß im DEFINE-Abschnitt mit VARIABLE oder PARAM definiert sein. Jeder Datentyp kann numerische Werte haben.

PAGENO
liefert als Zahl die aktuelle Seitennummer.

LINENO
liefert als Zahl die aktuelle Zeilennummer und beginnt die Zählung mit 1. Verwenden Sie LINENO nicht in PAGE HEADER oder PAGE TRAILER. LINENO wirkt dort nur auf der ersten Seite.

*-n-ausdr*
Negatives Vorzeichen für den Wert von *n-ausdr.* Das positive Vorzeichen '+' kann man auch weglassen.

*n-ausdr1 + n-ausdr2*
addiert *n-ausdr1* und *n-ausdr2.* Für die weiteren Operationen gilt:

- – subtrahiert *n-ausdr1* von *n-ausdr2,*
- * multipliziert *n-ausdr1* mit *n-ausdr2,*
- / dividiert *n-ausdr1* durch *n-ausdr2* und
- ** potenziert *n-ausdr1* mit *n-ausdr2.*

GROUP
ist nur bei der Ortsangabe AFTER GROUP erlaubt und bewirkt, daß die nachfolgenden Mengenfunktionen (COUNT usw.) sich nur auf diese Gruppe beziehen.

COUNT
liefert die Anzahl der Sätze, die die letzte Anweisung des SELECT-Abschnitts oder die READ-Anweisung liefert. Wenn GROUP vor COUNT steht, liefert COUNT nur die Anzahl der Sätze aus der letzten Gruppe. Wenn WHERE nach COUNT steht, liefert COUNT nur die Anzahl der Sätze, die die Bedingung erfüllen.

PERCENT
ist nur sinnvoll zusammen mit GROUP oder WHERE. PERCENT liefert den prozentualen Anteil der mit GROUP oder WHERE ausgewählten Sätze an allen Sätzen, die die letzte Anweisung des SELECT-Abschnitts oder die READ-Anweisung liefert. Das Ergebnis hat zwei Nachkommastellen (Datentyp DECIMAL mit 32 Stellen).

AVERAGE OF *ausdruck*
errechnet den Durchschnitt der Inhalte von *ausdruck* und liefert das Ergebnis (Datentyp DECIMAL mit 32 Stellen). Für jeden Satz, den die letzte Anweisung des SELECT-Abschnitts oder die READ-Anweisung liefert, wird einmal der Inhalt von *ausdruck* verwendet. Läßt sich *ausdruck* nicht als Zahl interpretieren, dann liefert die Mengenfunktion die Zahl 0 als Ergebnis. GROUP und WHERE schränken die Sätze wie bei COUNT ein. Sie erhalten ein unvorhergesehenes Ergebnis, wenn *ausdruck* eine vom Benutzer definierte Variable enthält (Variable definieren, siehe DEFINE-Abschnitt in diesem Kapitel).

AVG OF *ausdruck*
Abkürzung von AVERAGE.

MAX OF *ausdruck*
errechnet das Maximum der Inhalte von *ausdruck* und liefert das Ergebnis. Ansonsten gilt dasselbe wie bei AVERAGE.

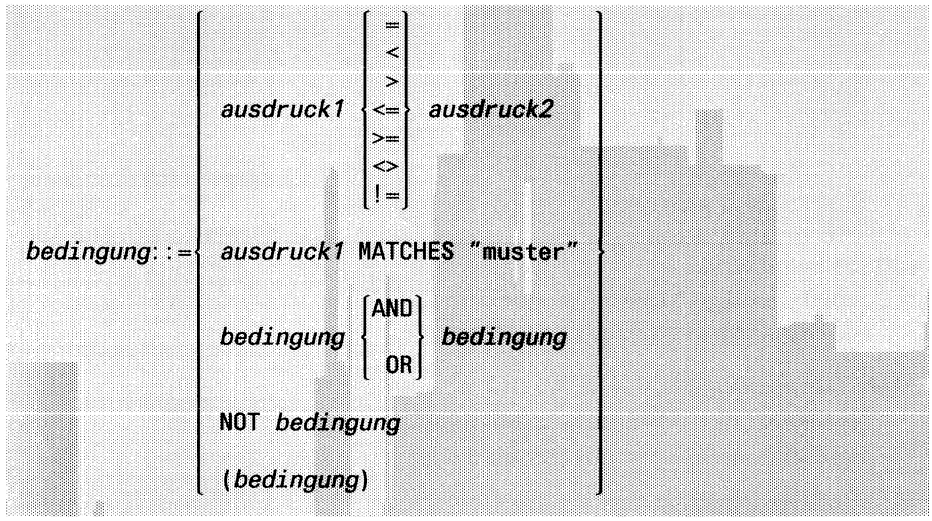
MIN OF *ausdruck*
errechnet das Minimum der Inhalte von *ausdruck* und liefert das Ergebnis. Ansonsten gilt dasselbe wie bei AVERAGE.

TOTAL OF *ausdruck*
summiert die Inhalte von *ausdruck* und liefert das Ergebnis. Ansonsten gilt dasselbe wie bei AVERAGE.

WHERE *bedingung*
bewirkt, daß sich die Mengenfunktionen (COUNT usw.) auf die Sätze beziehen, die *bedingung* erfüllen. Für den in *bedingung* enthaltenen Ausdruck gilt: Sie erhalten ein unvorhergesehenes Ergebnis, wenn *ausdruck* eine vom Benutzer definierte Variable enthält (Variable definieren, siehe DEFINE-Abschnitt in diesem Kapitel). Die WHERE-Bedingung darf sich nur auf Sätze beziehen, die in der SELECT- oder READ-Anweisung bereits definiert sind.

## Bedingungen für Anweisungen

Bedingungen kommen im FORMAT-Abschnitt bei der IF- und WHILE-Anweisung sowie bei WHERE für Mengenfunktionen vor.




*ausdruck1* = *ausdruck2*

Die Bedingung ist erfüllt, wenn *ausdruck1* den gleichen Inhalt wie *ausdruck2* hat. Bei den anderen Vergleichsoperatoren ist die Bedingung erfüllt:

| | |
|---|---|
| < | Inhalt *ausdruck1* kleiner als Inhalt *ausdruck2* |
| > | Inhalt *ausdruck1* größer als Inhalt *ausdruck2* |
| <= | Inhalt *ausdruck1* kleiner oder gleich dem Inhalt *ausdruck2* |
| >= | Inhalt *ausdruck1* größer oder gleich dem Inhalt *ausdruck2* |
| <> | Inhalt *ausdruck1* ungleich dem Inhalt *ausdruck2* |
| != | Inhalt *ausdruck1* ungleich dem Inhalt *ausdruck2* |

*ausdruck* MATCHES "muster"

*muster* ist eine alphanumerische Konstante, die in Anführungszeichen stehen muß.

Die Bedingung ist erfüllt, wenn der Inhalt von *ausdruck* zu *muster* paßt. Dies gilt in folgenden Fällen:

| **muster enthält** | **ausdruck enthält an derselben Stelle** |
|---|---|
| zeichen | genau das Zeichen |
| ? | ein beliebiges Zeichen |
| * | eine beliebige Zeichenfolge oder nichts |
| [ { zeichen / zeichen-zeichen / ^zeichen / ^zeichen-zeichen } ... ] | ein Zeichen, das zu den angegebenen Zeichen oder Zeichenbereichen gehört.<br>^ bezieht sich auf alle nachfolgenden Zeichen und gibt ihnen folgende Bedeutung: ein Zeichen enthält, das nicht zu den angegebenen Zeichen oder Zeichenbereichen gehört.<br>Die eckigen Klammern [] müssen Sie eingeben. |
| * | * |
| \? | ? |

*bedingung* AND *bedingung*
Die Bedingung ist erfüllt, wenn beide Bedingungen erfüllt sind.

*bedingung* OR *bedingung*
Die Bedingung ist erfüllt, wenn eine der beiden Bedingungen oder beide Bedingungen erfüllt sind.

NOT *bedingung*
Diese Bedingung ist erfüllt, wenn *bedingung* nicht erüllt ist.

*(bedingung)*
Die Klammern brauchen Sie, um Bedingungen zu einer Einheit zusammenzufassen (Rangfolge der Operatoren siehe Kolumnentitel FORMAT/Opteratoren).

# 6 Benutzermenüs



Benutzermenüs in INFORMIX sind Menüs, die Sie sich über das Menü BENUTZER-MENUE selbst definieren. Sie können sich damit eine Menüstruktur schaffen, die speziell auf Ihren Anwendungsfall zugeschnitten ist.

Ein selbst definiertes Benutzermenü kann z. B. folgendes Aussehen haben:

```
          Vertriebsgesellschaft Versand

  1. Formate
  2. Listen
  3. Abfragen
  4. Sonstiges


Funktion auswaehlen ueber Leertaste, Pfeiltaste oder Nummer.
Aktuelles Menue beenden mit 'e'.
Funktion aufrufen mit RETURN-Taste: ↵
```

**Leistungsumfang**

Je Datenbank können Sie genau eine Benutzermenüstruktur definieren.

Je Benutzermenü können Sie maximal 28 Menüfunktionen definieren.

Eine Menüfunktion kann eine der folgenden Aktionen ausführen:

- Aufruf eines Betriebssystemkommandos
- Aufruf einer SQL-Anweisung bzw. des INFORMIX-Menüs SQL-DIALOG
- Aufruf eines Formatprogrammes bzw. der Menüfunktion *Ablauf* des INFORMIX-Menüs FORMAT
- Aufruf eines Listenprogrammes bzw. der Menüfunktion *Ablauf* des INFORMIX-Menüs LISTE
- Aufruf aller Menüfunktionen eines Benutzermenüs

Die maximale Schachtelungstiefe von Menüaufrufen ist 19.

## 6.1 Definition eines Benutzermenüs

Zur Definition eines Benutzermenüs stellt Ihnen INFORMIX ein von PERFORM erzeugtes Format zur Verfügung, das Sie über entsprechende PERFORM-Menüfunktionen bedienen: Daten eingeben, suchen, ändern, löschen usw.

Sie erhalten dieses Eingabeformat, indem Sie die Funktion *Modifizieren* des Menüs BENUTZER-MENUE auswählen.

Die folgenden Abschnitte beschreiben:

- den Aufruf des Menüs BENUTZER-MENUE
- das Eingabeformat für die Benutzermenüdaten
- die Bedienung des Eingabeformats

### 6.1.1 Aufruf des Menüs BENUTZER-MENUE

Das Menü BENUTZER-MENUE können Sie sowohl über das Menüsystem als auch auf Betriebssystemebene aufrufen.

**Aufruf über das Menüsystem**

Wählen sie im Hauptmenü die Funktion *Benutzer-Menue* aus:

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  Tabelle  END
Ablauf oder Modifizieren eines vom Benutzer erstellten Menues
```

Wenn Sie noch keine Datenbank eröffnet haben, erscheint zunächst das Menü AUSWAHL DATENBANK. Dort wählen Sie eine Datenbank aus, in der die Daten für das Benutzermenü gespeichert werden sollen. Die Datenbank muß bereits vorhanden sein.

Anschließend erhalten Sie das Menü BENUTZER-MENUE; dort wählen Sie die Funktion *Modifizieren*:

```
BENUTZER-MENUE: Ablauf  Modifizieren  END
Modifizieren des Benutzer-Menues fuer die aktuelle Datenbank
                                  datenbank
```

**Aufruf auf Betriebssystemebene**

Kurzform:

```
isql datenbank -bm
```

**isql** ruft INFORMIX auf datenbank Datenbankname (muß bereits vorhanden sein) **-b** Funktion *Benutzer-Menue* im Hauptmenü **m** Funktion *Modifizieren* im Menü BENUTZER-MENUE

Der vollständige Aufruf ist in Kapitel 7.3.1 beschrieben.

### 6.1.2 Beschreibung des Eingabeformats

Die Funktion *Modifizieren* des Menüs BENUTZER-MENUE liefert folgendes Eingabeformat für Ihre Benutzermenü-Daten:

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                              (Tabelle   :         )

==========================MENUE-EINGABEFORMAT==============================

Menuename:  [                  ]

Menuetitel: [                                                           ]

-------------------------AUSWAHL-SEKTION---------------------------------

Auswahl-Nummer :   [       ]            Auswahl-Typ :    [ ]

Auswahl
Text:     [                                                             ]

Auswahl
Aktion:   [                                                             ]
```

Das Format besteht aus zwei Abschnitten:

Abschnitt MENUE-EINGABEFORMAT: Hier geben Sie Daten zur Definition von Menünamen ein. INFORMIX speichert die Daten in der Tabelle *sysmenus.*

Abschnitt AUSWAHL-SEKTION: Hier geben Sie Daten zur Definition von Menüfunktionen ein. INFORMIX speichert die Daten in der Tabelle *sysmenuitems.*

Die Tabellen *sysmenus* und *sysmenuitems* sind durch eine Master/Detail-Beziehung verbunden, wobei *sysmenus* als MASTER-Tabelle für *sysmenuitems* definiert ist. Gemeinsames Join-Feld für beide Tabellen ist das Feld *Menuename*:

| **Tabelle** | **Spalte** | **Bildschirmfeld** |
|---|---|---|
| sysmenus | *menuname* | *Menuename* |
| | title | Menuetitel |
| sysmenuitems | *imenuname* | *Menuename* |
| | itemnum | Auswahl-Nummer |
| | mtext | Auswahl Text |
| | mtype | Auswahl-Typ |
| | progname | Auswahl Aktion |

INFORMIX legt die Tabellen *sysmenus* und *sysmenuitems* an, sobald innerhalb der aktuellen Datenbank zum ersten Mal das Menü BENUTZER-MENUE mit der Funktion *Modifizieren* aufgerufen wird.

Die Anzahl der Sätze in *sysmenus* entspricht der Anzahl der definierten Menünamen; die Anzahl der Sätze in *sysmenuitems* entspricht der Anzahl aller definierten Menüfunktionen.

Eigentümer der Tabellen wird die aktuelle Benutzerkennung (Owner = userid). Standardmäßig sind folgende Zugriffsrechte definiert:

```
User     Select   Update   Insert    Delete   Index   Alter

public   All      All      Yes       Yes      Yes     No
```

### Bedeutung der Formatfelder

#### Feld Menuename - Menüname

Definiert den Menünamen, unter dem die Menüdefinition abzuspeichern ist. Diesen Menünamen müssen Sie im Feld *Auswahl Aktion* angeben, wenn Sie dort als Aktion den Aufruf dieses Menüs definieren wollen.

Für die Vergabe des Namens gelten die allgemeinen Namenskonventionen (siehe SQL-Handbuch [1]).

Für das oberste Menü müssen Sie immer den Menünamen *main* (in Kleinbuchstaben) definieren.

*Beispiel*

Der Menüname des obersten Menüs soll definiert werden:

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                              (Tabelle  1: sysmenus)

=========================MENUE-EINGABEFORMAT=============================

Menuename:  [main                ]

Menuetitel: [                                                          ]

-------------------------AUSWAHL-SEKTION---------------------------------

Auswahl-Nummer :                        Auswahl-Typ :

Auswahl
Text:

Auswahl
Aktion:

_________________________________________________________________________
```

### Feld Menuetitel - Menütitel

Definiert den Menütitel, wie er als Überschrift auf dem Benutzermenü-Bildschirm erscheint.

Die maximale Länge ist 60 Zeichen.

Ein Benutzermenü, das der Aktionsart S zugeordnet ist, erscheint nicht am Bildschirm (siehe Beschreibung des Feldes *Auswahl Aktion*).

*Beispiel*

Das oberste Menü (main) soll den Menütitel "Vertriebsgesellschaft Versand" erhalten:

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                              (Tabelle  1: sysmenus)

==========================MENUE-EINGABEFORMAT==============================

Menuename:  [main                ]

Menuetitel: [Vertriebsgesellschaft Versand                                ]

-------------------------AUSWAHL-SEKTION---------------------------------

Auswahl-Nummer :                          Auswahl-Typ :

Auswahl
Text:

Auswahl
Aktion:

_____________________________________________________________________________
```

### Feld Auswahl-Nummer - Auswahlnummer für die Menüfunktion

Auswahlnummer, die dem angegebenen Menüfunktionsnamen (Feld *Auswahl Text*) zugeordnet werden soll.

Sie können Ziffern von 1 bis 28 angeben. Maximal 28 Auswahlnummern (und damit Menüfunktionen) sind innerhalb eines Benutzermenüs möglich.

Der Aufbau des Benutzermenü-Bildschirms hängt ab von der Anzahl der Menüfunktionen (siehe Abschnitt 6.3).

*Anzeige der Auswahlnummer im Benutzermenü-Bildschirm*

Die Menüfunktionen werden nach Auswahlnummern aufsteigend sortiert. INFORMIX numeriert die Anzeige - beginnend mit 1 - lückenlos durch, unabhängig von den tatsächlich definierten Auswahlnummern.

Beispiel:

| **Definition Auswahlnummer** | **Name der Menüfunktion** | **Anzeige im Bildschirm** |
|---|---|---|
| 3 | a | 1 a |
| 4 | b | 2 b |
| 7 | c | 3 c |
| 11 | d | 4 d |

*Beispiel*

Für das Menü *main* wird die erste Menüfunktion definiert (*Auswahl-Nummer 1*):

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                          (Tabelle  2: sysmenuitems)

=========================MENUE-EINGABEFORMAT============================

Menuename:  main

Menuetitel:  Vertriebsgesellschaft Versand

-------------------------AUSWAHL-SEKTION--------------------------------

Auswahl-Nummer :   [1]            Auswahl-Typ :    [M]

Auswahl
Text:     [Formate                                                    ]

Auswahl
Aktion:   [format                                                     ]

_______________________________________________________________________
```

### Feld Auswahl-Typ - Aktionsart

Vereinbart die Aktionsart. Sie muß der im Feld *Auswahl Aktion* definierten Aktion entsprechen.

Sie geben jeweils den ersten Buchstaben der Aktionsart an. Kleinbuchstaben werden automatisch in Großbuchstaben umgewandelt. Folgende Aktionsarten sind möglich:

| **Aktionsart** | **Bedeutung/Aktion** |
|---|---|
| M(enü) | Aufruf eines Benutzermenüs |
| P(rogramm) | Betriebssystemaufruf (Programm oder Kommando) |
| A(bfragesprache) | Aufruf einer SQL-Anweisungsdatei |
| L(iste) | Aufruf eines Listenprogramms |
| F(ormat) | Aufruf eines Formatprogramms |
| S(cript) | Aufruf aller Menüfunktionen eines Benutzermenüs |

*Beispiel*

Die erste Menüfunktion des obersten Menüs soll ein Menüaufruf sein. Für den Auswahl-Typ wird daher *M* vereinbart.

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                          (Tabelle  2: sysmenuitems)

==========================MENUE-EINGABEFORMAT==============================

Menuename:  main

Menuetitel:  Vertriebsgesellschaft Versand

-------------------------AUSWAHL-SEKTION---------------------------------

Auswahl-Nummer :   [1      ]          Auswahl-Typ :    [M]

Auswahl
Text:     [Formate                                                    ]

Auswahl
Aktion:   [format                                                     ]
```

### Feld Auswahl Text - Menüfunktionsname

Definiert den Menüfunktionsnamen, wie er auf dem Bildschirm erscheint.

Die maximale Länge ist 60 Zeichen. Die Anzeige wird auf 33 Zeichen abgeschnitten, wenn das Benutzermenü mehr als 14 Menüfunktionen enthält (siehe Aufbau des Benutzermenü-Bildschirms, Abschnitt 6.3).

*Beispiel*

Die erste Menüfunktion soll den Namen "Formate" erhalten:

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                              (Tabelle  2: sysmenuitems)

=========================MENUE-EINGABEFORMAT==============================

Menuename:  main

Menuetitel:  Vertriebsgesellschaft Versand

-------------------------AUSWAHL-SEKTION---------------------------------

Auswahl-Nummer :   [1      ]          Auswahl-Typ :    [M]

Auswahl
Text:      [Formate                                                      ]

Auswahl
Aktion:    [format                                                       ]
```

**Feld Auswahl Aktion - Aktion**

Definiert die Aktion, die die Menüfunktion ausführen soll. Die Aktion muß der Aktionsart entsprechen, die Sie zuvor im Feld *Auswahl-Typ* definiert haben.

**Achtung:**

Die maximale Schachtelungstiefe von Menüaufrufen beträgt 19. Jeder weitere Menüaufruf ab Schachtelungstiefe 19 wird ignoriert.

| **Inhalt im Feld** *Auswahl-Typ* | **Eingabe in das Feld** *Auswahl Aktion* |
|---|---|
| M | Name eines Benutzermenüs, wie er im Feld *Menuename* definiert wurde. *Aktion:* Das angegebene Benutzermenü wird am Bildschirm angezeigt. |
| P | Betriebssystemaufruf: Name eines Programms oder Kommandos. *Aktion:* Das angegebene Programm oder Kommando wird ausgeführt, danach automatisch in das Benutzermenü zurückgekehrt. |
| A | Es gibt zwei Möglichkeiten für die Eingabe:<br>– Name einer SQL-Anweisungsdatei. Das Suffix *.sql* kann weggelassen werden. Die Datei muß im aktuellen Dateiverzeichnis vorhanden sein. *Aktion:* Die enthaltenen Anweisungen werden ausgeführt, danach automatisch in das Benutzermenü zurückgekehrt.<br>– keine Eingabe. *Aktion:* Das INFORMIX-Menü SQL-DIALOG wird aufgerufen. Sobald Sie das Menü beenden, wird automatisch in das Benutzermenü zurückgekehrt.<br>Wenn Sie innerhalb Ihrer SQL-Anweisungsfolge die Datenbank wechseln, so müssen Sie vor Beendigung der Anweisungsfolge wieder die Datenbank eröffnen, die das Benutzermenü enthält. Sonst ist die Rückkehr nicht möglich. |

L

Es gibt zwei Möglichkeiten für die Eingabe:

- Name eines Listenprogramms. Das Suffix *.arc* kann weggelassen werden. Die entsprechende Datei muß im aktuellen Dateiverzeichnis enthalten sein.
  *Aktion:* Das angegebene Listenprogramm wird ausgeführt, danach automatisch in das Benutzermenü zurückgekehrt.

- keine Eingabe.
  *Aktion:* Die Menüfunktion *Ablauf* des INFORMIX-Menüs LISTE wird aufgerufen. Sie geben den Namen des gewünschten Listenprogramms an. Das Listenprogramm wird ausgeführt, danach automatisch in das Benutzermenü zurückgekehrt.

Die Rückkehr in das Benutzermenü erfolgt auch dann automatisch, wenn sich ein Listenprogramm auf eine andere Datenbank bezieht.

F

Es gibt zwei Möglichkeiten für die Eingabe:

- Name eines Formatprogramms. Das Suffix *.frm* kann weggelassen werden. Die entsprechende Datei muß im aktuellen Dateiverzeichnis enthalten sein.
  *Aktion:* Das angegebene Formatprogramm wird ausgeführt, danach automatisch in das Benutzermenü zurückgekehrt.

- keine Eingabe.
  *Aktion:* Die Menüfunktion *Ablauf* des INFORMIX-Menüs FORMAT wird aufgerufen. Sie geben den Namen des gewünschten Formatprogramms an. Das Formatprogramm wird ausgeführt, danach automatisch in das Benutzermenü zurückgekehrt.

Die Rückkehr in das Benutzermenü erfolgt auch dann automatisch, wenn sich ein Formatprogramm auf eine andere Datenbank bezieht.

S — Name eines Benutzermenüs, wie er im Feld *Menuename* definiert wurde.
*Aktion:* Die Menüfunktionen des angegebenen Benutzermenüs werden sofort in definierter Reihenfolge (Feld *Auswahl-Nummer*) ausgeführt, ohne daß dieses Menü vorher am Bildschirm erscheint.

*Beispiel*

Die erste Menüfunktion soll als Aktion den Aufruf des Benutzermenüs *format* ausführen. Im Feld *Auswahl Aktion* wird daher der Menüname *format* angegeben.

```
PERFORM: Suchen Vorw. Rueckw. Blob Neuaufnehmen Korrigieren ...
                                          (Tabelle  2: sysmenuitems)

=========================MENUE-EINGABEFORMAT==============================

Menuename:  main

Menuetitel:  Vertriebsgesellschaft Versand

-------------------------AUSWAHL-SEKTION---------------------------------

Auswahl-Nummer :   [1      ]          Auswahl-Typ :    [M]

Auswahl
Text:      [Formate                                                    ]

Auswahl
Aktion:    [format                                                     ]
```

### 6.1.3 Bedienung des Eingabeformats

Die Bedienung des Eingabeformats setzt Kenntnisse im Umgang mit PERFORM voraus, insbesondere in der Bedienung des PERFORM-Bildschirms und in der Anwendung der PERFORM-Menüfunktionen (siehe dazu die entsprechenden Abschnitte der PERFORM-Beschreibung, Kapitel 4).

Dieser Abschnitt beschreibt lediglich den Ablauf, wie Sie anhand der PERFORM-Menüfunktionen Ihre Eingabe durchführen.

### Benutzermenü erzeugen

Ein Benutzermenü erzeugen, bedeutet, Daten in die Tabellen *sysmenüs* und *sysmenuitems* einzugeben. Dazu verwenden Sie das Eingabeformat.

| **Ablauf** | **PERFORM-Menüfunktion** |
|---|---|
| 1. Menüname und -titel definieren: Im ersten Formatabschnitt (MENUE-EINGABEFORMAT) definieren Sie Menüname und -titel. Das erzeugt einen Satz in der Master-Tabelle *sysmenus*. Diesen Vorgang wiederholen Sie, bis Sie alle Menünamen definiert haben. | Neuaufnehmen |
| 2. Benutzermenü auswählen: Wählen Sie das Benutzermenü aus, für das Sie die Menüfunktionen definieren wollen. | Suchen |
| 3. In Detail-Tabelle wechseln: Wechseln Sie in die Detail-Tabelle (AUSWAHL-SEKTION). PERFORM meldet, daß keine Sätze vorhanden sind, die mit dem Satz der Master-Tabelle (Join-Feld *Menuename*) verbunden sind. | Detail |
| 4. Menüfunktionen definieren: Definieren Sie Auswahlnummer, Aktionsart, Menüfunktionsname und Aktion für die erste Menüfunktion. Wiederholen Sie den Vorgang, bis Sie alle Menüfunktionen definiert haben. Für jede Menüfunktion wird ein Satz in *sysmenuitems* erzeugt. | Neuaufnehmen |
| 5. In Master-Tabelle wechseln: Wechseln Sie in die Master-Tabelle zurück und wiederholen Sie den gesamten Vorgang (Schritte 2 bis 5), bis Sie für alle Menüs Menüfunktionen definiert haben. | Master |

### Benutzermenü ändern und löschen

Das Ändern oder Löschen eines Benutzermenüs geschieht analog zum Erzeugen. Sie verwenden dazu die PERFORM-Menüfunktionen *Korrigieren* bzw. *Loeschen*.

## 6.2 Aufruf von Benutzermenüs

Benutzermenüs können Sie sowohl über das Menüsystem als auch auf Betriebssystemebene aufrufen.

**Aufruf über das Menüsystem**

Wählen Sie im Hauptmenü die Funktion *Benutzer-Menue* aus:

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  Tabelle  END
Ablauf oder Modifizieren eines vom Benutzer erstellten Menues
```

Wenn Sie noch keine Datenbank eröffnet haben, erscheint zunächst das Menü AUSWAHL DATENBANK. Dort wählen Sie die Datenbank aus, die die Benutzermenüs enthält.

Anschließend erhalten Sie das Menü BENUTZER-MENUE; dort wählen Sie die Funktion *Ablauf*:

```
BENUTZER-MENUE:  Ablauf  Modifizieren END
Ablauf des Benutzer-Menues fuer die aktuelle Datenbank
                                  datenbank
```

Anschließend erhalten Sie das oberste Benutzermenü (*main*), wenn Sie über die Umgebungsvariable DBMENU kein anderes Benutzermenü vereinbart haben (siehe Kapitel 7.2).

**Aufruf auf Betriebssystemebene**

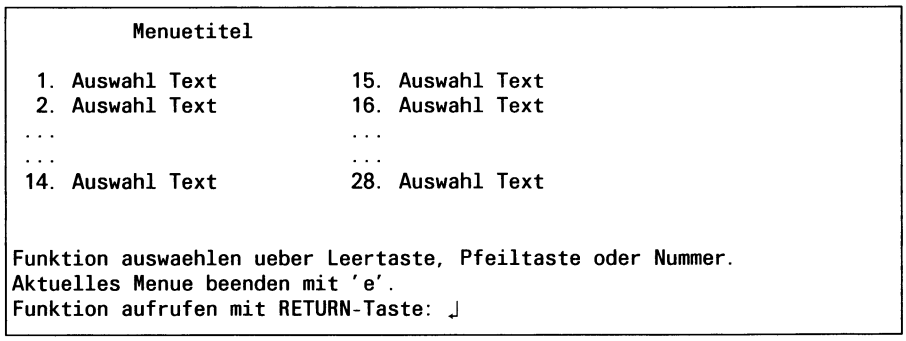
Kurzform:

```
isql␣datenbank␣-ba
```

| | |
|---|---|
| **isql** | ruft INFORMIX auf |
| datenbank | Name der Datenbank, die die Benutzermenüs enthält. |
| **-b** | Funktion *Benutzer-Menue* im Hauptmenü |
| **a** | Funktion *Ablauf* im Menü BENUTZER-MENUE |

Der vollständige Aufruf ist in Kapitel 7.3.1 beschrieben.

## 6.3 Aufbau und Bedienung des Benutzermenü-Bildschirms




**Bedeutung der Anzeigen im Bildschirm**

| | |
|---|---|
| Menuetitel | Menütitel, wie er im Feld *Menuetitel* definiert wurde. |
| 1. ... 28. | Auswahlnummer für die Menüfunktion. Sie muß nicht übereinstimmen mit der Nummer im Feld *Auswahl-Nummer* des Eingabeformats (siehe Abschnitt 6.1.2). Wenn mehr als 14 Menüfunktionen definiert sind, wird die Anzeige in zwei Spalten zu je 33 Zeichen aufgeteilt. Menüfunktionsnamen, die länger sind, werden abgeschnitten. |
| Auswahl Text | Name der Menüfunktion, wie er im Feld *Auswahl Text* definiert ist. |

**Bedienung des Bildschirms**

*Menüfunktion auswählen:* Geben Sie die Auswahlnummer ein oder markieren Sie die gewünschte Funktion, indem Sie die Leuchtmarke mittels Pfeil- oder Leertaste an die entsprechende Stelle bewegen.

*Menüfunktion aufrufen:* Drücken Sie die Taste [↵].

*Menü beenden:* Drücken Sie die Taste [END] oder geben Sie 'e' ein. Auf oberster Menüebene bedeutet das: Rückkehr in das INFORMIX-Menü BENUTZER-MENUE.

# 7 Systemumgebung von INFORMIX



## 7.1 INFORMIX-Dateien

INFORMIX erzeugt und verwaltet für jede Datenbank Dateien, die Informationen über Format- und Listenprogramme sowie Anweisungsdateien enthalten. Die Dateien werden standardmäßig im aktuellen Dateiverzeichnis gespeichert.

Bei INFORMIX-SE werden außerdem die Tabellen über Dateien verwaltet. Dazu legt INFORMIX-SE für jede Datenbank ein Dateiverzeichnis an, das die Tabellen-Dateien enthält.

Jeder von INFORMIX erzeugte Datei-Eintrag endet mit einem bestimmten Suffix, dessen Bedeutung nachfolgend erklärt ist.

### Dateien für Formatprogramme, Listenprogramme und Anweisungsdateien

| Datei | | Bedeutung |
|---|---|---|
| Name | Suffix | |
| format | .per | format = vom Benutzer definierter Formatname; Formatprogramm, das die Definition für ein Format (Bildschirmmaske) enthält. Der Benutzer kann das Programm automatisch (über FORMBUILD) oder selbst (über Editor) erzeugen. Damit das dort enthaltene Format verwendet werden kann, muß das Formatprogramm erst übersetzt werden. (FORMBUILD-Funktion). |
| | .frm | Übersetztes Formatprogramm. Diese Datei wird von PERFORM verwendet. |
| liste | .ace | liste = vom Benutzer definierter Listenname; Listenprogramm, das die Definition für ein Ausgabeformat (Liste) enthält. Der Benutzer kann das Programm automatisch (über ACEPREP) oder selbst (über Editor) erzeugen. Damit das dort enthaltene Ausgabeformat verwendet werden kann, muß das Listenprogramm erst übersetzt werden (ACEPREP-Funktion). |
| | .arc | Übersetztes Listenprogramm. Diese Datei wird von ACEGO verwendet. |
| anweisung | .sql | anweisung = vom Benutzer definierter Anweisungsdateiname; enthält SQL-Anweisungen zum Bearbeiten der Datenbank, die häufig benötigt werden. Der Benutzer kann diese Datei automatisch (über SQL) oder selbst (über Editor) erzeugen. |

## Dateien für Datenbanktabellen (nur INFORMIX-SE)

<table>
<tr><th colspan="2">Dateiverzeichnis</th><th rowspan="2" colspan="3">Bedeutung</th></tr>
<tr><th>Name</th><th>Suffix</th></tr>
<tr><td>datenbank</td><td>.dbs</td><td colspan="3">datenbank = vom Benutzer definierter Datenbankname;<br>Dateiverzeichnis für die Datenbank. Es enthält Informationen über Tabellen, die der Benutzer erstellt hat, sowie über Systemtabellen, die INFORMIX automatisch anlegt.<br>Folgende Dateien sind enthalten:</td></tr>
<tr><td></td><td></td><td colspan="2">Datei</td><td rowspan="2">Bedeutung</td></tr>
<tr><td></td><td></td><td>Name</td><td>Suffix</td></tr>
<tr><td></td><td></td><td>tabnnn</td><td></td><td>tab = vom Benutzer definierter Tabellenname; maximal 7-stellig. Ist der Name kürzer, so werden Unterstriche '_' ergänzt.<br>nnn = von INFORMIX vergebene Nummer.<br>Abweichend davon kann der Benutzer in der Anweisung CREATE TABLE (siehe SQL-Handuch) für die Datei<br>- einen anderen Namen bestimmen und auch<br>- ein anderes Dateiverzeichnis</td></tr>
<tr><td></td><td></td><td>sysmenus<br>sysmenuitems</td><td></td><td>Tabellen für Benutzermenüs; werden nur bei Bedarf angelegt.</td></tr>
<tr><td></td><td></td><td>syscolauth<br>syscolumns<br>sysdepend<br>sysindexes<br>syssynonyms<br>systabauth<br>systables<br>sysusers<br>sysviews<br>syssyntable<br>sysconstraints</td><td></td><td>sys... = Namen von Systemtabellen, die INFORMIX automatisch anlegt. Der Aufbau dieser Tabellen ist im SQL-Handbuch [1] näher erläutert.</td></tr>
<tr><td></td><td></td><td></td><td>.dat</td><td>Datei, die die Daten der bezeichneten Tabelle enthält.</td></tr>
<tr><td></td><td></td><td></td><td>.idx</td><td>Datei, die Index-Informationen über die bezeichnete Tabelle enthält.</td></tr>
</table>

**Beispiel**

Für eine INFORMIX-SE-Datenbank *test1* sind im aktuellen Dateiverzeichnis folgende Dateien gespeichert (die Ausgabe ist zur Demonstration nach Suffix sortiert):

```
$ll | sort -t. +1 -2 +0.41 -1

insgesamt 26 KB
-rw------- 1 lomata      988 Jun  4 14:54 clist1.ace     }
-rw------- 1 lomata     1282 Jun  4 14:54 clist2.ace     } Listenprogramme

-rw------- 1 lomata      613 Jun  4 15:03 clist1.arc     } übersetzte
-rw------- 1 lomata      850 Jun  4 15:03 clist2.arc     } Listenprogramme

drwx--x--x 2 lomata      480 Jun  4 15:08 test1.dbs      Datenbank-
                                                         Dateiverzeichnis
-rw------- 1 lomata     2138 Jun  4 14:54 auftrag.frm    }
-rw------- 1 lomata     3172 Jun  4 14:54 best.frm       } übersetzte
-rw------- 1 lomata     1006 Jun  4 14:54 kunde.frm      } Formatprogramme

-rw------- 1 lomata     1610 Jun  4 14:54 auftrag.per    }
-rw------- 1 lomata     2995 Jun  4 14:54 best.per       } Formatprogramme
-rw------- 1 lomata      791 Jun  4 14:54 kunde.per      }

-rw------- 1 lomata      151 Jun  4 14:54 ex1.sql        }
-rw------- 1 lomata      168 Jun  4 14:54 ex2.sql        }
-rw------- 1 lomata      100 Jun  4 14:54 ex3.sql        } Anweisungsdateien
-rw------- 1 lomata      101 Jun  4 14:54 ex4.sql        }
```

## 7.2 Umgebungsvariablen

Sie können die "Umgebung" beeinflussen, in der INFORMIX arbeitet. Dies geschieht über Variablen, deren Inhalt Sie individuell verändern können und die INFORMIX anschließend entsprechend auswertet. Die folgende Tabelle faßt alle von INFORMIX verwendeten Umgebungsvariablen kurz zusammen:

| Umgebungsvariable | Bedeutung |
|---|---|
| DBANSIWARN | Überwachung des ANSI-Standards |
| DBDATE | Ein- und Ausgabeformat für das Datum |
| DBDELIMITER | Trennzeichen definieren |
| DBEDIT | Texteditor bestimmen |
| DBMENU | Standard-Benutzermenü vereinbaren |
| DBMONEY | Ausgabeformat für Geldbeträge |
| DBPATH | Pfade für Datenbanken und Dateien |
| DBPRINT | Ausgabeprogramm definieren |
| DBTEMP | Dateiverzeichnis für temporäre Dateien |
| DBLANG | Dateiverzeichnis für Meldungsdateien |
| INFORMIXDIR | Dateiverzeichnis für INFORMIX-Dateien |
| INFORMIXTERM | Bildschirmsteuerung termcap/terminfo |
| SQLEXEC | INFORMIX-Backend bestimmen |
| TBCONFIG | *tbconfig*-Datei für das INFORMIX-ONLINE-System |

Ist eine Umgebungsvariable nicht gesetzt, so benutzt INFORMIX einen Standardwert.

### Setzen von Umgebungsvariablen

Umgebungsvariablen setzen Sie in der Shell. Beachten Sie, daß Sie Umgebungsvariablen in Großbuchstaben angeben müssen.

Beispiel (Bourne-Shell):

```
DBTEMP=/abc
export DBTEMP
```

Beispiel (C-Shell):

```
setenv DBTEMP /abc
```

Wenn Sie diese Eingaben nicht nach jedem LOGIN neu vornehmen wollen, dann müssen Sie sie in der Datei

**.profile** (Bourne-Shell),
**.login** (C-Shell) )

speichern.

Mit *unset* (bzw. *unsetenv*) können Sie die Variablen zurücksetzen.

### Abfragen von Umgebungsvariablen

`echo␣$variable`
gibt den Wert aus, auf den die angegebene Variable *variable* gesetzt ist.

`set`
gibt alle Variablen aus, die für die aktuelle Shell gesetzt sind.

`export`
gibt die Namen aller exportierten Variablen aus.

### Umgebungsvariable DBANSIWARN

Mit DBANSIWARN überprüfen Sie die Erweiterungen von INFORMIX gegenüber dem ANSI-Standard. DBANSIWARN brauchen Sie keinen Wert zuweisen. Sie müssen sie lediglich explizit setzen.
Die Funktion entspricht der des Schalters **-ansi**, der in bestimmten Programmen verwendet werden kann. Die Warnungen werden folgendermaßen ausgegeben:

- Bei *isql* erfolgt die Ausgabe im Menu.
- Bei *saceprep* werden die Warnungen in die Datei mit Suffix *.err* geschrieben.

### Umgebungsvariable DBDATE

definiert das Ein- und Ausgabeformat eines Datums.

*Erlaubte Werte:*

Der Wert von DBDATE besteht aus den Buchstaben **D**, **M** und **Y** in beliebiger Reihenfolge. Hinter dem Y muß eine 2 oder 4stehen. Der Wert muß mit Punkt '.' oder Schrägstrich '/' oder Bindestrich '-' abgeschlossen sein.

Die Reihenfolge von D, M und Y bestimmt die Reihenfolge der Tages-, Monats- und Jahreszahl im Datum. Dabei bedeutet:

D eine zweistellige Tageszahl
M eine zweistellige Monatszahl
Y2 eine zweistellige Jahreszahl
Y4 eine vierstellige Jahreszahl; (zweistelligen Angaben werden automatisch in vierstellige umgewandelt: 19xx).

Das letzte Zeichen des Wertes ist das Trennzeichen, das die Tages-, Monats- und Jahreszahlen im Datum trennt.

Standardwert: `DBDATE=DMY4.`

Das bedeutet: Als Reihenfolge gilt Tag, Monat, Jahr; die Jahreszahl kann 4-stellig eingegeben werden; die Angaben Tag, Monat und Jahr sind durch das Zeichen '.' (Punkt) zu trennen.

**Beispiel**

- DBDATE = DMY4. (Standardwert)

  Ausgabe: 17.09.1990

- DBDATE = MDY2

  Ausgabe: 09-17-90

### Umgebungsvariable DBDELIMITER

vereinbart ein Zeichen, das in Ein- oder Ausgabedateien als Trennzeichen zwischen den einzelnen Feldern eines Datensatzes interpretiert werden soll. Das Trennzeichen wirkt sich auf folgende Dateien aus:

- Eingabedateien, die die SQL-Anweisung LOAD verarbeitet
- Eingabedateien, die der ACE-Abschnitt READ verarbeitet
- Ausgabedateien, die die SQL-Anweisung UNLOAD erzeugt
- Ausgabedateien, die die PERFORM-Funktion PRINT erzeugt

*Standardwert:* `DBDELIMITER ist |`

| = senkrechter Strich (ASCII-Code:124)

**Umgebungsvariable DBEDIT**

vereinbart, welcher Editor für die Bearbeitung von Formaten, Listen und Anweisungen verwendet wird.

Standardwert: `DBEDIT=ced`

Das bedeutet: INFORMIX verwendet den Editor **ced**.

**Umgebungsvariable DBMENU**

Mit DBMENU bestimmen Sie das Benutzermenü, das mit der Funktion *Ablauf* des Menüs BENUTZER-MENUE als erstes Menü aufgerufen werden soll.

Standardwert: DBMENU = main

**Beispiel**

```
DBMENU=formate
```

Als erstes Benutzermenü wird das Menü *formate* aufgerufen.

**Umgebungsvariable DBLANG**

bestimmt das Dateiverzeichnis für fremdsprachliche Meldungstexte.

Das angegebene Dateiverzeichnis muß in dem Dateiverzeichnis enthalten sein, das die Umgebungsvariable INFORMIXDIR bezeichnet. Dort ist standardmäßig /usr/lib/informix definiert.

Standardwert: `DBLANG=msg`

Das Standard-Dateiverzeichnis für Meldungstexte heißt *msg*.
Wenn Sie ein anderes Meldungsdateiverzeichnis als *msg* verwenden wollen, so müssen Sie dieses Dateiverzeichnis unter $INFORMIXDIR einrichten. Die Zugriffsrechte sind foldendermaßen zu setzen:

USER und GROUP: informix
Lese-, Schreib-, Ausführberechtigung: 755

Die zum Meldungsdateiverzeichnis gehörenden *.iem*-Dateien müssen Sie unter das Dateiverzeichnis $INFORMIXDIR/$DBLANG kopieren. Diese Dateien benötigen folgende Zugriffsrechte:

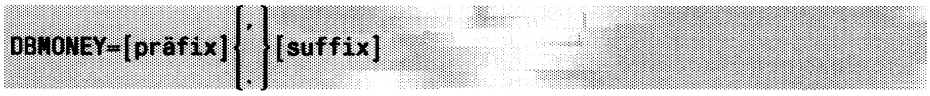
USER und GROUP: informix
Lese-, Schreib-, Ausführberechtigung: 644

### Umgebungsvariable DBMONEY

legt für die Ausgabe von Werten

- in DECIMAL- und MONEY-Spalten fest, welches Zeichen zwischen Vor- und Nachkommastellen gesetzt wird: Komma oder Punkt. Dies gilt unabhängig davon, ob die Eingabe mit Komma oder Punkt erfolgt.
- in MONEY-Spalten ein Präfix und/oder ein Suffix fest, das zusammen mit dem Spaltenwert ausgegeben wird.

*Syntax:*




*Standardwert:* DBMONEY=,

präfix
Zeichenfolge, die am Anfang des Wertes ausgegeben wird.
Max. Länge: 7 Zeichen
Verbotene Zeichen: Zahlen, Komma, Punkt.

suffix
Zeichenfolge, die am Ende des Wertes ausgegeben wird. Länge und Zeichenvorrat wie bei *präfix.*

, Zahlen mit Nachkommastellen erhalten (unabhängig von der Eingabe) bei der Ausgabe ein Komma als Trennzeichen.

. Zahlen mit Nachkommastellen erhalten (unabhängig von der Eingabe) bei der Ausgabe einen Punkt als Trennzeichen.

**Beispiel**

- DBMONEY=, (Standardwert)

  Ausgabe: 4768,36

- DBMONEY=.Dollar

  Ausgabe: 4768.36 Dollar

### Umgebungsvariable DBPATH

legt Pfade zu Dateiverzeichnissen fest, die INFORMIX zusätzlich zum aktuellen Dateiverzeichnis durchsuchen soll. DBPATH wird nur ausgewertet, wenn INFORMIX im aktuellen Dateiverzeichnis nicht fündig wird und zwar

- beim Suchen der Datenbank (nur INFORMIX-SE)
- beim Suchen eines Formats
- beim Suchen einer Liste

Für INFORMIX-STAR gibt die Umgebungsvariable DBPATH an, in welchem Informixsystem die Datenbank gesucht wird.

Die einzelnen Pfade sind mit Doppelpunkt (:) zu trennen.

Standardmäßig sucht INFORMIX nur im aktuellen Dateiverzeichnis.

**Beispiel**

```
DBPATH=/usr/a:/usr/b:/usr/c
```

Wird die Datenbank im aktuellen Dateiverzeichnis nicht gefunden, durchsucht INFORMIX die Dateiverzeichnisse /usr/a, /usr/b und /usr/c.

### Umgebungsvariable DBPRINT

bestimmt das Ausgabeprogramm

- für Listen (bei OUTPUT TO PRINTER).
- für die Funktion PRINT im Menü SQL-DIALOG.

Standardwert: `DBPRINT=lpr`

Listen werden mit dem Kommando *lpr* ausgedruckt.

### Umgebungsvariable DBTEMP

bestimmt das Dateiverzeichnis, das die temporären Dateien aufnimmt.

Standardwert: `DBTEMP=/tmp`

Temporäre Dateien werden im Dateiverzeichnis /tmp abgelegt.

### Umgebungsvariable INFORMIXDIR

bestimmt interne Dateiverzeichnisse von INFORMIX.

Standardwert: `INFORMIXDIR=/usr/lib/informix`

Die INFORMIX-Dateien sind unter /usr/lib/informix abgelegt.

### Umgebungsvariable INFORMIXTERM

Mit INFORMIXTERM können Sie bestimmen, ob INFORMIX *termcap* oder *terminfo* zur Bildschirmsteuerung auswertet. Wenn Sie INFORMIXTERM nicht setzen, gilt automatisch *termcap.*

Beispiel: `INFORMIXTERM=termcap`

INFORMIX wertet die Datei termcap aus dem Dateiverzeichnis /usr/lib/informix aus.

### Umgebungsvariable SQLEXEC

Die Umgebungsvariable SQLEXEC muß nur dann gesetzt werden, wenn INFORMIX-SE und INFORMIX-ONLINE auf dem gleichen Rechner installiert sind und Sie Zugriff auf INFORMIX-SE haben wollen.

Beispiel: `SQLEXEC=$INFORMIXDIR/lib/sqlexec`

Ist SQLEXEC nicht gesetzt, ist INFORMIX-ONLINE der Standard.

Standardwert: `SQLEXEC=$INFORMIXDIR/lib/sqlturbo`

### Umgebungsvariable TBCONFIG

Die Umgebungsvariable TBCONFIG gibt das INFORMIX-ONLINE-System an, wenn mehrere Systeme auf einem Rechner initialisiert sind.

Standardwert: `$INFORMIXDIR/etc/tbconfig`

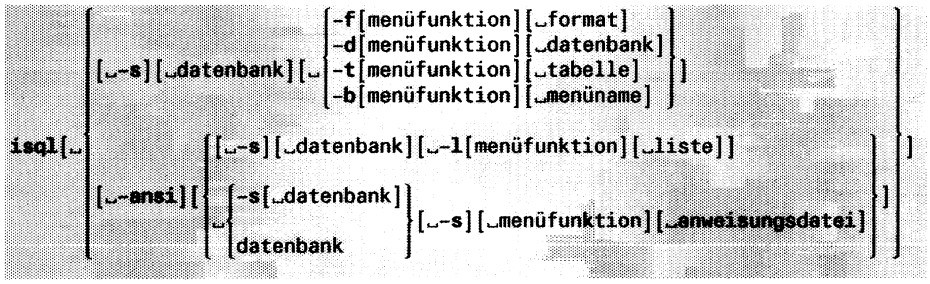
## 7.3 Programmaufrufe

Dieser Abschnitt faßt mit Ausnahme der Dienstprogramme (siehe Anhang A.2) sämtliche Programmaufrufe zusammen, die Ihnen auf Betriebssystemebene zur Verfügung stehen:

| | |
|---|---|
| isql | INFORMIX mit oder ohne Menüs aufrufen |
| sformbld | Format-Quellprogramm übersetzen |
| sperform | Format-Quellprogramm ablaufen lassen |
| saceprep | Listen-Quellprogramm übersetzen |
| sacego | Listenprogramm ablaufen lassen |

### 7.3.1 isql - INFORMIX mit Menüs aufrufen

Mit isql rufen Sie INFORMIX auf. Ohne weitere Angaben erhalten Sie anschließend das Hauptmenü. Sie können mit einem *isql*-Aufruf aber auch sofort in ein ausgewähltes Menü verzweigen und dort Funktionen ausführen. INFORMIX wird automatisch beendet, sobald Sie dieses Menü wieder verlassen.




**isql**
ruft INFORMIX auf.

**-s**
unterdrückt die Ausgabe von Meldungen (silent Option).Wichtige Meldungen werden dennoch ausgegeben.

**-ansi**
prüft, ob die SQL-Anweisungen Erweiterungen gegenüber dem ANSI-Standard enthalten und gibt dann eine Warnung aus.

datenbank
Geben Sie die Datenbank an, die Sie eröffnen wollen. Fehlt die Angabe, so fordert INFORMIX den Datenbanknamen später automatisch an.

Für INFORMIX-SE gilt: INFORMIX sucht das Dateiverzeichnis der Datenbank (datenbank.dbs) im aktuellen Dateiverzeichnis. Ist es dort nicht vorhanden, so wird die Umgebungsvariable DBPATH ausgewertet (Umgebungsvariablen, siehe Kapitel 7.2). Sie können anstelle der Datenbank auch einen Pfadnamen angeben. INFORMIX durchsucht dann nur den angegebenen Pfad. Das Suffix *.dbs* dürfen Sie nicht angeben.

**-f** wählt das Menü FORMAT aus.

menüfunktion
Wählen Sie die gewünschte Funktion des Menüs FORMAT aus (1 Buchstabe). Folgende Angaben sind sinnvoll:

a = Ablauf
m = Modifizieren
g = Generieren
n = Neu
c = Compilieren
l = Löschen

format
Name des gewünschten Formatprogramms. Das Suffix geben Sie nicht an.

**-l** wählt das Menü LISTE aus.

menüfunktion
Wählen Sie die gewünschte Funktion des Menüs LISTE aus (1 Buchstabe). Folgende Angaben sind sinnvoll:

a = Ablauf
m = Modifizieren
g = Generieren
n = Neu
c = Compilieren
l = Löschen

liste
Name des gewünschten Listenprogramms. Das Suffix geben Sie nicht an.

**-d** wählt das Menü DATENBANK aus.

menüfunktion
Wählen Sie die gewünschte Funktion des Menüs DATENBANK aus (1 Buchstabe). Folgende Angaben sind sinnvoll:

n = Neu
l = Löschen

datenbank
Name der gewünschten Datenbank, auf die sich die angegebene *menüfunktion* bezieht. Ist die Menüfunktion **l** (löschen) angegeben, so wird hier die Angabe *datenbank* ignoriert, d.h. in das Datenbank-Auswahl-Menü verzweigt.

**-t** wählt das Menü TABELLE aus.

menüfunktion
Wählen Sie die gewünschte Funktion des Menüs TABELLE aus (1 Buchstabe). Folgende Angaben sind sinnvoll:

n = Neu
m = Modifizieren
i = Info
l = Löschen

tabelle
Name der gewünschten Tabelle, auf die sich die angebene *menüfunktion* bezieht. Ist die Menüfunktion **n** angegeben, so wird die Angabe *tabelle* ignoriert, d.h. in das Tabellen-Auswahl-Menü verzweigt.

**-b** wählt das Menü BENUTZER-MENUE aus.

menüfunktion
Wählen Sie die gewünschte Funktion des Menüs BENUTZER-MENUE aus (1 Buchstabe). Folgende Angaben sind sinnvoll:

a = Ablauf
m = Modifizieren

menüname
Name des gewünschten Benutzermenüs.

**-s** wählt das Menü SQL-DIALOG aus.
Fehlt die Angabe *datenbank*, so fordert INFORMIX den Datenbanknamen an, bevor das Menü ausgegeben wird. Wenn Sie dort auch keinen Datenbanknamen eingeben, so wird keine Datenbank eröffnet. Dann können Sie anschließend nur eine der Anweisungen CREATE DATABASE, DATABASE, DROP DATABASE oder START DATABASE ausführen bzw. die Anweisungsdatei muß mit einer von diesen Anweisungen beginnen.

menüfunktion
Wählen Sie die gewünschte Funktion des Menüs SQL-DIALOG aus (1 Buchstabe). Lediglich folgende Angabe ist sinnvoll:

i = Info

anweisungsdatei
Name der gewünschten Anweisungsdatei. Der Dateinamen muß mit dem Suffix *.sql* enden; Sie geben jedoch das Suffix nicht mit an.

### 7.3.2 isql - INFORMIX ohne Menüs aufrufen

Bei diesem Aufruf rufen Sie INFORMIX ohne Menüs auf. Dabei können Sie eine Anweisungdatei angeben, die anschließend ausgeführt wird. Sonst geben Sie ihre Anweisungen direkt nach dem Aufruf ein. Mit der Taste [END] beenden Sie INFORMIX.


**isql**
ruft INFORMIX auf.

**-ansi**
prüft, ob die SQL-Anweisungen Erweiterungen gegenüber dem ANSI-Standard enthalten und gibt dann eine Warnung aus.

- Der Bindestrich anstelle eines Datenbank-Namens bewirkt, daß keine Datenbank eröffnet wird. Dann können Sie anschließend nur eine der Anweisungen CREATE DATABASE, DATABASE, DROP DATABASE oder START DATABASE ausführen bzw. die Anweisungsdatei muß mit einer von diesen Anweisungen beginnen.

datenbank

Geben Sie die zu eröffnende Datenbank an. Für INFORMIX-SE gilt: INFORMIX sucht das Dateiverzeichnis der Datenbank (datenbank.dbs) im aktuellen Dateiverzeichnis. Ist es dort nicht vorhanden, so wird die Umgebungsvariable DBPATH ausgewertet (Umgebungsvariablen, siehe Kapitel 7.2). Sie können anstelle der Datenbank auch einen Pfadnamen angeben. INFORMIX durchsucht dann nur den angegebenen Pfad. Das Suffix *.dbs* dürfen Sie nicht angeben.

- Der Bindestrich '-' weist *stdin* als Eingabequelle für die Anweisungen zu, d.h. INFORMIX liest aus *stdin* die Anweisungen. Wenn *stdin* die Tastatur ist, fordert INFORMIX mit dem Zeichen '>' die erste SQL-Anweisung an. Sie können anschließend mehrere SQL-Anweisungen eingeben; jede einzelne muß mit Semikolon ';' abgeschlossen sein. Mit der Taste END beenden Sie INFORMIX.

anweisungsdatei

Geben Sie den Namen einer Anweisungsdatei an, deren Anweisungen Sie ausführen möchten. Der Dateiname muß mit dem Suffix *.sql* enden; Sie geben jedoch das Suffix nicht mit an. Die einzelnen Anweisungen in der Datei müssen durch ein Semikolon ';' getrennt sein.

### 7.3.3 sformbld - Format-Quellprogramm übersetzen

```
sformbld[␣-q][␣-l␣zeilen][␣-c␣spalten][␣-v]{␣datei
                                            ␣-d}
```

**-q** unterdrückt alle Meldungen am Bildschirm.

**-l** zeilen
Maximale Seitenlänge innerhalb des SCREEN-Abschnitts in Zeilen. Davon bleiben 4 Zeilen für das System reserviert. Fehlt die Angabe, so gilt als Standardwert: 24 Zeilen.

**-c** spalten
Maximale Breite des Formats in Spalten. Fehlt die Angabe, so gilt als Standardwert: Spaltenanzahl der längsten Zeile im SCREEN-Abschnitt.

**-v** vergleicht bei Spalten vom Datentyp CHAR die Länge der im SCREEN-Abschnitt definierten Bildschirmfelder mit der Länge der jeweils zugeordneten Tabellenspalten.

Falls Längenunterschiede auftreten, werden Warnungen in eine Datei *datei.err* ausgegeben. Diese Datei *datei.err* dient lediglich zu Ihrer Information über aufgetretene Längenunterschiede. Das Format ist benutzbar, obwohl Längenunterschiede aufgetreten sind. Sie können allerdings durch diese Längenunterschiede Schwierigkeiten bei der Ein- und Ausgabe von Daten bekommen (siehe dazu Kapitel 3.3, SCREEN-Abschnitt, *feldbegrenzer*).

Falls Sie Längenunterschiede beseitigen wollen, müssen Sie das in der Datei *datei.per* tun. Die Datei *datei.per* müssen Sie nach einer Korrektur erneut übersetzen. Bei erfolgreicher Übersetzung wird die Datei *datei.err* gelöscht.

**-d** erfüllt zwei Aufgaben: Zum einen wird ein Quellprogramm für ein Standard-Format erzeugt. Gleichzeitig wird dieses Quellprogramm übersetzt und so ein ablauffähiges Formatprogramm erzeugt. Die hierfür notwendigen Informationen fragt FORMBUILD im Dialog vom Benutzer ab. FORMBUILD fragt nach dem Namen des zu erstellenden Standard-Formatprogramms, nach dem Namen der Datenbank und nach den Tabellen, die Sie verwenden wollen. Sie dürfen bei diesem Schalter maximal 14 Tabellen angeben. Sie können das Formatprogramm aber auf bis zu 16 Tabellen erweitern.

Als Seitenlänge im SCREEN-Abschnitt generiert FORMBUILD automatisch SCREEN SIZE 20 (siehe SCREEN-Abschnitt im Kapitel 3.3).

datei

muß der Name der Datei sein, die das Format-Quellprogramm enthält. Der Dateiname hat die Endung *.per*, die Sie hier aber nicht anzugeben brauchen.

FORMBUILD übersetzt das Format-Quellprogramm und schreibt das übersetzte Formatprogramm in eine Datei mit dem Suffix *.frm*. Auf diese Datei greift PERFORM zu, wenn ein Format aufgerufen wird.

Falls beim Übersetzen Fehler aufgetreten sind, legt *sformbld* eine Fehlerdatei mit dem Suffix *.err* an. Diese Datei enthält eine Kopie des Quellprogramms mit Fehlerkommentaren. Sie müssen dann die Fehler im Quellprogramm korrigieren und das Programm erneut compilieren. Bei erfolgreicher Übersetzung wird die Fehlerdatei gelöscht.

### 7.3.4 sperform - Formatprogramm ablaufen lassen

```
sperform datei...
```

datei
ist der Name des übersetzten Formatprogramms. Die Datei, die das ablauffähige Formatprogramm enthält, hat das Suffix *.frm*. Das Suffix *.frm* geben Sie nicht an.

PERFORM sucht die Datei im aktuellen Dateiverzeichnis. Wenn es sie dort nicht findet, wertet es die Umgebungsvariable DBPATH aus (siehe Kapitel 7.2). DBPATH muß dann anzeigen, wo sich die Datei befindet.

Wenn Sie mehrere Dateien angeben, müssen Sie diese durch Leerzeichen voneinander trennen. PERFORM gibt die Formate entsprechend der im Programmaufruf genannten Reihenfolge aus. Wenn PERFORM ein Format nicht anzeigen kann, wird abgebrochen. Eventuell nachfolgende Formate können dann ebenfalls nicht mehr angezeigt werden.

### 7.3.5 saceprep Listen-Quellprogramm übersetzen

```
saceprep[ -q][ -ansi][ -o dateiverzeichnis] datei
```

**-q** unterdrückt alle Meldungen am Bildschirm, die keine Fehlermeldungen sind.

**-ansi**
prüft, ob die SQL-Anweisungen im SELECT-Abschnitt Erweiterungen gegenüber dem ANSI-Standard enthalten. ACEPREP gibt dann eine Warnung aus.

**-o** dateiverzeichnis
schreibt die Dateien, die *saceprep* erzeugt, ins angegebene Dateiverzeichnis.

Standardmäßig kommen die Dateien in das Dateiverzeichnis, in dem das Quellprogramm steht.

datei
ist der Name der Datei, die das Listen-Quellprogramm enthält. Er muß das Suffix *.ace* haben, aber Sie brauchen das Suffix nicht anzugeben.

### 7.3.6 sacego Listenprogramm ablaufen lassen

```
sacego[␣-q][␣-d␣datenbank]␣datei[␣parameterwerte]...
```

**-q** unterdrückt alle Meldungen am Bildschirm, die keine Fehlermeldungen sind.

**-d** datenbank
legt eine andere Datenbank fest, für die die Liste erstellt werden soll. Die Liste muß natürlich zu dieser Datenbank passen.

Standardmäßig erstellt *sacego* die Liste für die Datenbank, die im Listen-Quellprogramm steht (siehe Kapitel 5.3, DATABASE-Abschnitt).

datei
ist der Name der Datei, die das ablauffähige Listenprogramm enthält. Er muß das Suffix *.arc* haben, aber Sie brauchen das Suffix nicht anzugeben. Sie können auch die Namen von mehren Listenprogrammen angeben. Diese laufen dann nacheinander ab.

parameterwerte
Wenn für das Listenprogramm Parameter definiert sind (siehe Kapitel 5.3, DEFINE-Abschnitt), dann muß man hier die Werte in der richtigen Reihenfolge hinschreiben. Zwischen zwei Werten steht immer ein Leerzeichen.

## 7.4 Einträge in der TERMCAP-Datei

INFORMIX ist unabhängig von einem speziellen Bildschirm oder einer speziellen Tastatur. Die Informationen zum Bildschirm und zur Tastatur holt sich ein Programm aus der TERMCAP-Datei. Diese ist standardmäßig die Datei /etc/termcap.

Sie können also INFORMIX auf Ihre individuellen Bedürfnisse anpassen, indem Sie Veränderungen bzgl. der TERMCAP-Datei vornehmen. Es gibt hierfür mehrere Möglichkeiten.

**Achtung: Es können auch andere Programme die Felder benutzen, die INFORMIX benutzt. Änderungen betreffen alle Benutzer!**

**Hinweis**

INFORMIX verwendet eine eigene TERMCAP-Datei /usr/lib/informix/termcap. Verwenden Sie nur INFORMIX-Produkte, versorgen Sie die Umgebungsvariable TERMCAP folgendermaßen:

```
TERMCAP=/usr/lib/informix/termcap
```

Verwenden Sie auch andere Produkte, die TERMCAP auswerten, belegen Sie die Umgebungsvariable INFORMIXTERM:

```
INFORMIXTERM=termcap
```

Damit wird die Datei /usr/lib/informix/termcap ausgewertet.

In den folgenden Abschnitten erfahren Sie

- welche Felder aus der TERMCAP-Datei INFORMIX verwendet,
- wie Sie die Datei etc/termcap ändern oder eine eigene TERMCAP-Datei verwenden können und
- welche besonderen Möglichkeiten INFORMIX für die Kombination von Bildschirm-Attributen bietet (invertierte, halbhelle usw. Darstellung).

Eine ausführliche Beschreibung der Syntax für eine TERMCAP-Datei finden Sie im CES-Handbuch [14].

**Hinweis**

Programme, die zeilenorientiert arbeiten und nur an der Position der Schreibmarke Ausgaben machen, verwenden die Felder der TERMCAP-Datei nicht.

### 7.4.1 Von INFORMIX verwendete Felder in der TERMCAP-Datei

In der TERMCAP-Datei verwendet man drei unterschiedliche Typen von Feldern:

Typ 1
: Boolsche Felder: ist das entsprechened Feld vorhanden, dann ist auch die Bildschirmfunktion vorhanden (Beispiel: ...:am:...).

Typ 2
: Numerische Felder: nach dem Feld steht '#' und dann eine Zahl. Diese ist der Wert, den das Feld hat (Beispiel: ...:li#24:...).

Typ 3
: Zuweisungsfelder: nach dem Feld steht ein '=' und danach kommt eine Zeichenfolge. Diese ist der Wert, den das Feld hat (Beispiel: ...:ce=\E6K:...).

  Die Gesamtanzahl der Bytes in den Zeichenfolgen ist begrenzt:

  - Bei den Feldern für die Tastatur darf die Gesamtanzahl maximal 300 - Felderanzahl sein.
  - Bei den Feldern für den Bildschirm darf die Gesamtanzahl maximal 512 - Felderanzahl sein.

## Liste der Felder für die Tastatur

| Feld | Typ | Bedeutung |
|---|---|---|
| kl | 3 | ← |
| kr | 3 | → |
| kd | 3 | ↓ |
| ku | 3 | ↑ |
| kh | 3 | ↖ |
| ta | 3 | ⇥ |
| bt | 3 | ⇤ |
| k0 | 3 | CHAR INSERT |
| k1 | 3 | CHAR DELETE |
| k2 | 3 | WORD INSERT |
| k3 | 3 | WORD DELETE |
| k4 | 3 | LINE INSERT |
| k5 | 3 | LINE DELETE |
| k6 | 3 | |
| k7 | 3 | |
| k8 | 3 | ⇩ |
| k9 | 3 | ⇧ |
| l0 | 3 | HELP |
| l1 | 3 | START |
| l2 | 3 | END |
| P1-P9 | 3 | F1 - F9 |
| Pa-Pk | 3 | F10 - F20 |
| Pm, Pn | 3 | F21 , F22 |
| F1-F9 | 3 | F23 - F31 ( F23 : SHIFT F1 ) |
| Fa-Fm | 3 | F32 - F44 ( ab F37 nur SINIX) |
| Pl | 3 | PRINT |

**Liste der Felder für den Bildschirm**

| Feld | Typ | Bedeutung |
|---|---|---|
| li | 2 | Anzahl der Bildschirmzeilen |
| co | 2 | Anzahl der Bildschirmspalten |
| am | 1 | Schreibmarke springt am Zeilenende in neue Zeile |
| cm | 3 | freie Positionierung der Schreibmarke am Bildschirm |
| up | 3 | Schreibmarke eine Zeile nach oben |
| do | 3 | Schreibmarke eine Zeile nach unten |
| ce | 3 | Löschen der Zeile ab Schreibmarkenposition |
| cl | 3 | Löschen des gesamten Bildschirms |
| ti | 3 | Bildschirm-Initialisierung bei Programmstart |
| te | 3 | Wenn vorhanden, wird die Zeichenfolge bei Programmende ausgegeben. |
| vb | 3 | Wenn nicht vorhanden, wird ^G ausgegeben (akustisches Signal). |
| so | 3 | Anfang *invertiert*, wenn ZA nicht vorhanden |
| se | 3 | Ende *invertiert*, wenn ZA nicht vorhanden |
| as | 3 | Anfang *halbhell*, wenn ZA nicht vorhanden |
| ae | 3 | Ende *halbhell*, wenn ZA nicht vorhanden |
| us | 3 | Anfang *unterstrichen*, wenn ZA nicht vorhanden |
| ue | 3 | Ende *unterstrichen*, wenn ZA nicht vorhanden |
| bo | 3 | Anfang *blinkend*, wenn ZA nicht vorhanden |
| be | 3 | Ende *blinkend*, wenn ZA nicht vorhanden |

Fortsetzung der Tabelle auf der nächsten Seite

Liste der Felder für Bildschirm (Fortsetzung)

| Feld | Typ | Bedeutung |
|---|---|---|
| GS | 3 | Grafikmodus einschalten. Wenn GE nicht vorhanden, dann wird GS ignoriert: Linien mit '-' usw. |
| GE | 3 | Grafikmodus ausschalten. |
| GB | 3 | Wird nur ausgewertet, wenn GS ausgewertet wird. Definition der Grafikzeichen. Die Zeichenkette muß genau sechs Zeichen lang sein. Jede Position in der Zeichenkette legt eine Grafikdarstellung fest:<br>1. Position: Zeichen für linke obere Ecke<br>2. Position: Zeichen für linke untere Ecke<br>3. Position: Zeichen für rechte obere Ecke<br>4. Position: Zeichen für rechte untere Ecke<br>5. Position: Zeichen für waagerechte Linie<br>6. Position: Zeichen für senkrechte Linie<br>Ist GB nicht definiert dann gilt als Standardwert "BDCEA®" (Semigrafik-Zeichen für Terminal 97801). |
| ZA | | Sonderfeld für INFORMIX. Aus der angegeben Folge von Anweisungen baut INFORMIX eine Zeichenkette für die Bildschirm-Attribute zusammen. Dadurch werden Kombinationen von Attributen möglich (z.B. invertiert und blinkend). Die genaue Beschreibung enthält Abschnitt 7.4.3. |

### 7.4.2 Veränderungen über die TERMCAP-Datei

Die Veränderungen können Sie auf unterschiedliche Weise vornehmen.

- Wenn Sie auf dem SINIX-Rechner Systemverwalter-Rechte haben, dann können Sie die Feld-Einträge in der Datei /etc/termcap ändern. Sie müssen in dem Abschnitt ändern, den die Umgebungsvariable TERM beim Aufruf von INFORMIX bezeichnet. TERM=97801 bezeichnet z.B. die Stelle, wo die Datensichtstation 97801 beschrieben ist.
  Die Werte, die Sie eintragen, finden Sie in einer Schnittstellen-Beschreibung für Ihre Datensichtstation.

  **Achtung: Es können auch andere Programme die Felder benutzen, die INFORMIX benutzt. Änderungen betreffen alle Benutzer!**

- Als normaler Benutzer mit Shell-Erlaubnis können Sie auch jede andere Datei als TERMCAP-Datei verwenden: Sie belegen dazu die SINIX-Umgebungsvariable TERMCAP mit dem Namen der Datei.
  Eine eigene TERMCAP-Datei bekommen Sie beispielsweise, wenn Sie

  – die Datei /etc/termcap in eines Ihrer Dateiverzeichnisse kopieren,

  – die kopierte Datei so ändern, wie es der Systemverwalter in der Datei /etc/termcap tut und

  – TERMCAP mit dem Namen der veränderten Datei belegen und exportieren.

**Hinweis**

Die Datei /usr/lib/informix/termcap ist für INFORMIX und die Datensichtstation 97801 geschrieben. Sie enthält auch das Feld ZA.

### 7.4.3 ZA - Feld für Bildschirm-Attribute

Wenn ein INFORMIX-Programm ein Bildschirm-Attribut (invertiert, halbhell usw.) setzt, dann wertet es in der TERMCAP-Datei das Feld ZA aus.
Wenn das Feld nicht vorhanden ist, dann sind die Bildschirm-Attribute im Programm ohne Sinn.

Im Feld ZA steht eine Folge von Anweisung, die ein Programm ausführt, wenn es Bildschirm-Attribute setzt.
Man benutzt in den Anweisungen die Parameter p1 bis p4. Deren Werte liefert das Programm. Sie haben bei INFORMIX folgende Bedeutung:

| Parameter | Wert | Darstellung |
|---|---|---|
| p1 | 0 | WHITE oder NORMAL |
| | 1 | YELLOW |
| | 2 | MAGENTA |
| | 3 | RED oder BOLD |
| | 4 | CYAN |
| | 5 | GREEN |
| | 6 | BLUE oder DIM |
| | 7 | BLACK oder INVISIBLE |
| p2 | 0 | nicht invertiert |
| | 1 | invertiert |
| p3 | 0 | nicht blinkend |
| | 1 | blinkend |
| p4 | 0 | nicht unterstrichen |
| | 1 | unterstrichen |

Die Belegung *p1=6, p2=1, p3=0, p4=1* heißt beispielsweise, daß eine Darstellung halbhell, invertiert und unterstrichen sein soll.

Zunächst ist es aber ganz unerheblich, was in den Parametern *p1* bis *p4* steht. Erst die Anweisungsfolge im ZA-Feld wertet die Parameter aus und veranlaßt die Ausgabe der ensprechenden Steuerzeichen am Bildschirm.

### Die Anweisungen im ZA-Feld

Die Anweisungen für das ZA-Feld arbeiten mit einem Stack. Dabei ist der Stack ein Speicherbereich, der nach dem Prinzip *Last in - First out* organisiert ist:

- Holt eine Anweisung ein Element vom Stack, dann ist dieses immer das zuletzt auf den Stack gelegte Element. Die dahinter liegenden Elemente rücken nach.
- Legt eine Anweisung ein Element auf den Stack, dann schiebt sie damit alle Elemente um eins zurück, die schon auf dem Stack sind. Maximal kann der Stack zehn Elemente aufnehmen.

| Anweisung | Wirkung |
|---|---|
| %% | Gibt "%" am Bildschirm aus. |
| beliebige Zeichenfolge ohne "%" am Anfang | Gibt das angegebene Zeichen am Bildschirm aus. ESC wird mit "\E" angegeben, CTRL mit "^". |
| %d | Holt ein Element vom Stack und schreibt es wie *printf* in C auf den Bildschirm. |
| %2d | Wirkt entsprechend %d. |
| %3d | Wirkt entsprechend %d. |
| %c | Wirkt entsprechend %d. |
| %s | Wirkt entsprechend %d. |
| %l | Holt eine Zeichenkette vom Stack und legt deren Länge auf den Stack (Anzahl der Zeichen). |

| Anweisung | Wirkung |
|---|---|
| %p1 - %p9 | Legt den Inhalt vom entsprechenden Parameter (*p1* bis *p9*) auf den Stack. |
| %Pa - %Pz | Holt ein Element vom Stack und speichert es für die Zeit der Auswertung von ZA in der entsprechenden Variablen (*Pa* bis *Pz*). |
| %ga - %gz | Liest die entsprechende Variable (*Pa* bis *Pz*) und legt den Inhalt auf den Stack. |
| %' c' | Legt das konstante Zeichen *c* auf den Stack. |
| %{n} | Legt die ganze Zahl *n* auf den Stack. |
| %Sa - %Sz | Holt ein Element vom Stack und speichert es für die Zeit des Programmlaufs in der entsprechenden Variablen (*Sa* bis *Sz*). |
| %Ga - %Gz | Liest die entsprechende Variable (*Sa* bis *Sz*) und legt den Inhalt auf den Stack. |
| %+ %- %* %/ %m | Holt zwei Elemente vom Stack und verknüpft beide mit der jeweiligen arithmetischen Operation (%m entspricht *modulo*). Das Ergebnis legt die Anweisung wieder auf den Stack. |
| %& %\| %^ %~ | Holt zwei Elemente vom Stack und verknüpft beide mit der jeweiligen Bit-Operation:<br>%& *und*<br>%\| *oder*<br>%^ *exklusives oder*<br>%~ *bitweises Komplement*<br>Das Ergebnis legt die Anweisung wieder auf den Stack. |
| %= %> %< | Holt zwei Elemente vom Stack und verknüpft beide mit dem jeweiligen Vergleichsoperator. Wenn der Vergleich wahr ist, dann ist das Ergebnis ungleich "0"; ist er unwahr, dann ist es "0". Die Anweisung legt das Ergebnis auf den Stack. |
| %! %~ | Holt ein Element vom Stack, wendet die Operation an und schreibt das Ergebnis wieder auf den Stack.<br>%! Ergebnis einer Vergleichsoperation *negieren*<br>%~ *invertieren* |
| %? bedingung<br>%t then-Zweig<br>%e else-Zweig | Falls *bedingung* ungleich "0" ist, wird in den *then*-Zweig verzweigt, bei Gleichheit in den *else*-Zweig. Dieser ist wahlfrei.<br>Man kann die Anweisung im *else*-Zweig schachteln.<br>Man schreibt z. B. *b1* usw. für *bedingung*:<br>%?b1%t...%eb2%t...%eb3%t...%e...%; |

**Zur Schreibweise im ZA-Feld**

- Zwischen den Anweisungen dürfen keine Leerzeichen stehen.
- Die Anweisungen müssen in einer Zeile stehen.
- Die Zeichenkette, die mit dem ZA-Feld aufgebaut wird, kann maximal 50 Zeichen lang sein.

**Beispiel**

Für die Datensichtstation 97801 kann das Feld ZA folgendermaßen definiert sein (eigentlich alles in einer Zeile):

```
:ZA=\E[%?%p1%{3}%>%p1%{7}%<%&%t2%e%p1%{7}%=%t8%e0%;%?%p4%t;4%;
    %?%p3%t;5%;%?%p2%t;7%;m:
```

Erklärung:

```
\E[                 Schreibe ESC[ am Bildschirm.
%?%p1%{3}%>         Falls p1 größer 3 und kleiner 7 ist,
   %p1%{7}%<%&
   %t2                 schreibe 2 am Bildschirm (halbhell).
   %e%p1%{7}%=         Sonst, falls p1 gleich 7 ist,
      %t8                 schreibe 8 (unsichtbar).
      %t0                 Sonst schreibe 0 (normal).
%;                  Ende von %?.
%?%p4               Falls p4 ungleich 0,
   %t;4                schreibe ;4 (unterstrichen).
%;                  Ende von %?.
%?%p3               Falls p3 ungleich 0,
   %t;5                schreibe ;5 (blinkend).
%;                  Ende von %?.
%?%p2               Falls p2 ungleich 0,
   %t;7                schreibe ;7 (invertiert).
%;                  Ende von %?.
m                   Schreibe m am Bildschirm (Ende der ESCAPE-Folge).
```

## 7.5 Einträge in der TERMINFO

Wenn Sie die Umgebungsvariable INFROMIXTERM auf *terminfo* gesetzt haben, wertet INFORMIX zur Terminalsteuerung die Informationen aus dem Dateiverzeichnis /usr/lib/terminfo aus. Damit können Sie

- Funktionstasten definieren
- Bildschirmattribute bestimmen
- graphische Zeichen für die Darstellung von Formaträndern festlegen

Allerdings können Sie mit *terminfo* keine Farben bestimmen. Wollen Sie Farbattribute festlegen, müssen Sie die Datei *termcap* benutzen.

**Hinweis**

Dem Softwareentwickler steht eine Bibliothek zur Bildschirmsteuerung zur Verfügung (libcurses.a). Die Bibliothek arbeitet mit binären Beschreibungsdateien. Das Dateiverzeichnis /usr/lib/terminfo enthält die compilierten Beschreibungen der Bedieneinheiten. Die binäre Beschreibungsdatei für die Bedieneinheit 97801 hat den Pfadnamen /usr/lib/terminfo/9/97801.

Detaillierte Informationen über das Format der compilierten Beschreibungsdatei finden Sie im CES-Handbuch [14] und im Schnittstellen-Handbuch [15].

# A Anhang



## A.1 Beispieldatenbank versand

In diesem Abschnitt wird die Beispieldatenbank *versand* beschrieben, die für die Beispiele in diesem Handbuch verwendet wird. Die Struktur der Datenbank-Tabellen werden beschrieben, für welche Spalten ein Index definiert ist und über welche Spalten eine Verbindung der Tabellen (Join) möglich ist.

Für jede Tabelle werden die in ihr enthaltenen Daten aufgelistet.

### Struktur der Tabellen

Die Datenbank *versand* enthält Informationen über einen fiktiven Sportartikelgroßhändler, der Läden in Süddeutschland beliefert. Die Datenbank besteht aus fünf Tabellen:

- kunde
- auftrag
- posten
- artikel
- hersteller
- bundesland

#### Tabelle kunde

Die Tabelle *kunde* enthält Informationen über 18 Läden, die Sportartikel vom Großhändler beziehen. Für jeden Laden wird der Name des Ladeninhabers und der Name und die Adresse des Ladens gespeichert. Die Tabelle *kunde* hat folgende Spalten:

| Spaltenname | Datentyp |
|---|---|
| kunden_nr | SERIAL |
| vorname | CHAR(15) |
| nachname | CHAR(15) |
| firma | CHAR(20) |
| adresse1 | CHAR(20) |
| adresse2 | CHAR(20) |
| ort | CHAR(15) |
| bundesland | CHAR(2) |
| plz | CHAR(5) |
| telefon | CHAR(18) |

Für die Spalte *kunden_nr* ist ein eindeutiger Index definiert. Für die Spalte *plz* ist ebenfalls ein Index definiert; bei diesem sind doppelt vorkommende Werte erlaubt.

### Tabelle auftrag

Die Tabelle *auftrag* enthält die Aufträge, die die Kunden dem Großhändler erteilt haben. Für jeden Auftrag ist die Nummer des Auftrags gespeichert; zusätzlich das Auftragsdatum, die Kundennummer, Versandanweisungen, evtl. Rückstände, die Nummer des Kundenbestellscheins, Versandkosten und das Datum, an dem der Kunde für den Auftrag bezahlt hat. Die Spalten in der Tabelle *auftrag* sind:

| Spaltenname | Datentyp |
|---|---|
| auftrags_nr | SERIAL |
| auftragsdatum | DATE |
| kunden_nr | INTEGER |
| lieferhinweis | CHAR(40) |
| offen | CHAR(1) |
| fremd_nr | CHAR(10) |
| lieferdatum | DATE |
| liefergewicht | DECIMAL(8,2) |
| zustellgebuehr | MONEY(6,2) |
| zahldatum | DATE |

Für die Spalte *auftrags_nr* ist ein eindeutiger Index definiert; die Spalte *kunden_nr* ist ebenfalls indiziert, läßt aber doppelt vorkommende Werte zu.

### Tabelle posten

Die Tabelle *posten* enthält die einzelnen Positionen eines Auftrags. Beispiel: Einige Aufträge enthalten nur einen Posten, während andere Aufträge fünf Posten haben. Die Tabelle *posten* enthält die Postennummer, Auftragsnummer, Artikelnummer, Herstellercode, Menge und den Gesamtpreis für jeden Artikel, der bestellt wurde. Die Spalten in der Tabelle *posten* sind:

| Spaltenname | Datentyp |
|---|---|
| posten_nr | SMALLINT |
| auftrags_nr | INTEGER |
| artikel_nr | SMALLINT |
| herstellercode | CHAR(3) |
| menge | SMALLINT |
| gesamtpreis | MONEY(8,2) |

Die Spalte *auftrags_nr* ist indiziert und läßt doppelt vorkommende Werte zu. Die Spalten *artikel_nr* und *herstellercode* bilden einen zusammengesetzen Index, doppelt vorkommende Werte sind erlaubt.

### Tabelle artikel

Der Großhändler bietet seinen Kunden fünfzehn verschiedene Arten von Sportartikeln an. Beispiel: Der Großhändler bietet Ski-Handschuhe von drei verschiedenen Herstellern und Basketbälle von einem Hersteller an.

Die Tabelle *artikel* ist ein Katalog der Artikel, die vom Großhändler verkauft werden. Für jeden Artikel im Lager hat die Tabelle *artikel* eine Nummer, die den Artikel kennzeichnet, einen Code, der den Hersteller bezeichnet, eine Beschreibung des Artikels, den Preis der Packung, die Mindestmenge, die bestellt werden muß und eine Beschreibung der Pakkung (z.B.: Eine Kiste enthält zehn Fußbälle).

Die Tabelle *artikel* enthält folgende Spalten:

| Spaltenname | Datentyp |
|---|---|
| artikel_nr | SMALLINT |
| herstellercode | CHAR(3) |
| bezeichnung | CHAR(15) |
| preis | MONEY(6,2) |
| liefereinheit | CHAR(4) |
| stueckliste | CHAR(15) |

Die Spalten *artikel_nr* und *herstellercode* bilden einen zusammengesetzten Index, der eindeutige Werte enthalten muß.

### Tabelle hersteller

Der Großhändler hat Sportartikel von fünf Herstellern. Informationen über diese Hersteller sind in der Tabelle *hersteller* gespeichert. Diese Informationen bestehen aus einem Herstellercode und dem Namen des Herstellers. Die Spalten der Tabelle *hersteller* sind:

| Spaltenname | Datentyp |
|---|---|
| herstellercode | CHAR(3) |
| herstellername | CHAR(15) |

Für die Spalte *herstellercode* ist ein eindeutiger Index definiert.

### Tabelle bundesland

Die Tabelle *bundesland* enthält die Namen und Abkürzungen der 11 Bundesländer der Bundesrepublik Deutschland. Die Spalten der Tabelle *bundesland* sind:

| Spaltenname | Datentyp |
|---|---|
| bcode | CHAR(2) |
| laendername | CHAR(20) |

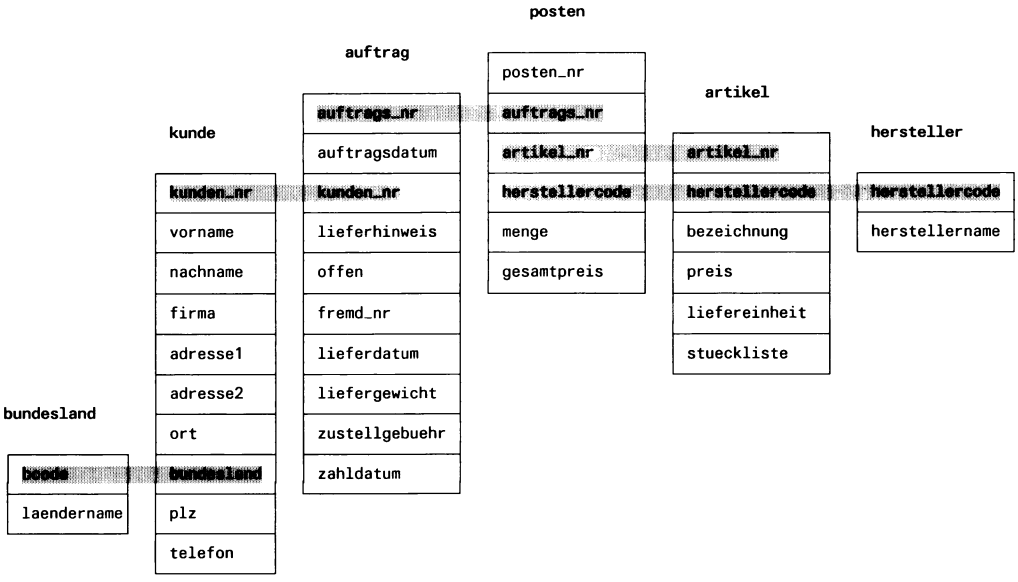
Für die Spalte *bcode* ist ein eindeutiger Index definiert.

## Verbindung der Tabellen durch Join-Spalten

Die sechs Tabellen der Datenbank *versand* lassen sich durch Spalten verbinden, die als Join-Spalten geeignet sind. Das ermöglicht eine Datenbankabfrage über mehrere Tabellen gleichzeitig.

### Übersicht

Die gerasterten Spalten zeigen die sinnvollen Join-Verbindungen zwischen den Tabellen der Datenbank.




Im folgenden wird jeweils die Verbindung von zwei Tabellen betrachtet.

### Join-Spalten in den Tabellen kunde und auftrag

Die Tabellen *kunde* und *auftrag* lassen sich durch die Spalte *kunden_nr* in der Tabelle *kunde* und *kunden_nr* in der Tabelle *auftrag* verbinden.

Ausschnitt aus der Tabelle *kunde*:

| kunden_nr | vorname | nachname |
|---|---|---|
| 101 | Ludwig | Pauli |
| 102 | Karola | Sadler |
| 103 | Philipp | Korres |
| 104 | Anton | Hochfeld |

Ausschnitt aus der Tabelle *auftrag*:

| auftrags_nr | auftragsdatum | kunden_nr |
|---|---|---|
| 1001 | 20. 01. 1990 | 104 |
| 1002 | 01. 06. 1990 | 101 |
| 1003 | 12. 10. 1990 | 104 |
| 1004 | 12. 04. 1990 | 106 |

Die Tabelle *kunde* enthält in der Spalte *kunden_nr* für jeden Kunden eine eindeutige Nummer und zusätzlich Spalten für Namen, Firma, Adresse, Telefonnummer etc. Die Tabelle *auftrag* enthält ebenfalls eine Spalte *kunden_nr*, die die Nummer des Kunden enthält, der den Auftrag gab und weitere Spalten, die den Auftrag betreffen.

Will man die Aufträge von Anton Hochfeld abfragen, so verbindet man die Tabellen *kunde* und *auftrag* über die Spalte *kunden_nr*. Anton Hochfeld hat die Kundennummer 104 (aus Tabelle *kunde* gelesen). In der Tabelle *auftrag* gibt es zwei Aufträge mit der Kundennummer 104, Anton Hochfeld hat also zur Zeit zwei Aufträge gegeben.

Durch die Verbindung der Tabellen *kunde* und *auftrag* über die Join-Spalte *kunden_nr* kann der Name und die Adresse eines Kunden gleichzeitig mit den zugehörigen Aufträgen abgefragt werden.

### Join-Spalten in den Tabellen auftrag und posten

Die Tabellen *auftrag* und *posten* lassen sich durch die Spalte *auftrags_nr* in der Tabelle *auftrag* und *auftrags_nr* in der Tabelle *posten* verbinden. In der Tabelle *auftrag* wird jeder Auftrag eines Kunden durch eine eindeutige Auftragsnummer identifiziert. Für jeden Posten dieses Auftrags wird ein Satz mit der zugehörigen Auftragsnummer und Informationen über den bestellten Artikel in die Tabelle *posten* geschrieben.

Durch die Join-Spalte *auftrags_nr* wird die Zuordnung zwischen Auftrag und zugehörigen Posten hergestellt.

Ausschnitt aus der Tabelle *auftrag*:

| auftrags_nr | auftragsdatum | kunden_nr |
|---|---|---|
| 1001 | 20. 01. 1990 | 104 |
| 1002 | 01. 06. 1990 | 101 |
| 1003 | 12. 10. 1990 | 104 |

Ausschnitt aus der Tabelle *posten*:

| posten_nr | auftrags_nr | artikel_nr | herstellercode |
|---|---|---|---|
| 1 | 1001 | 1 | HRO |
| 1 | 1002 | 4 | HSK |
| 2 | 1002 | 3 | HSK |
| 1 | 1003 | 9 | ANZ |
| 2 | 1003 | 8 | ANZ |
| 3 | 1003 | 5 | ANZ |

### Join-Spalten in den Tabellen posten und artikel

Die Tabellen *posten* und *artikel* lassen sich durch zwei Spalten verbinden: *artikel_nr* und *herstellercode* in der Tabelle *posten* und *artikel_nr* und *herstellercode* in der Tabelle *artikel.*

Beide Spalten werden zur eindeutigen Identifizierung eines Artikels benötigt; z.B. ist der Artikel mit der Artikelnummer 1 und dem Herstellercode HRO ein Herold Ski-Handschuh, während der Artikel mit der Artikelnummer 1 und dem Herstellercode HSK ein Hasken Ski-Handschuh ist.

Die gleichen Artikelnummern und Herstellercodes können in mehreren Sätzen der Tabelle *posten* auftreten, wenn dieser Artikel in verschiedenen Aufträgen bestellt wurde.

Durch diese beiden Spalten *artikel_nr* und *herstellercode* wird der Zusammenhang zwischen Posten eines Auftrags ( = bestellter Artikel) in der Tabelle *posten* und dem Artikel in der Tabelle *artikel* hergestellt.

Ausschnitt aus der Tabelle *posten*:

| posten_nr | auftrags_nr | artikel_nr | herstellercode |
|---|---|---|---|
| 1 | 1001 | 1 | HRO |
| 1 | 1002 | 4 | HSK |
| 2 | 1002 | 3 | HSK |
| 1 | 1003 | 9 | ANZ |
| 2 | 1003 | 8 | ANZ |
| 3 | 1003 | 5 | ANZ |
| 1 | 1004 | 1 | HRO |

Ausschnitt aus der Tabelle *artikel*:

| artikel_nr | herstellercode | bezeichnung |
|---|---|---|
| 1 | HRO | Ski-Handschuhe |
| 1 | HSK | Ski-Handschuhe |
| 1 | SMT | Ski-Handschuhe |

### Join-Spalten in den Tabellen artikel und hersteller

Die Tabellen *artikel* und *hersteller* lassen sich durch die Spalte *herstellercode* in der Tabelle *artikel* und *herstellercode* in der Tabelle *hersteller* verbinden. Dieser Herstellercode erscheint in mehreren Sätzen der Tabelle *artikel*, in der Tabelle *hersteller* ist er eindeutig.

Durch die Verbindung der Tabellen *artikel* und *hersteller* über die Spalte *herstellercode* kann zu einem bestimmten Artikel der volle Name des Herstellers gefunden werden.

Ausschnitt aus der Tabelle *artikel*:

| artikel_nr | herstellercode | bezeichnung |
|---|---|---|
| 1 | HRO | Ski-Handschuhe |
| 1 | HSK | Ski-Handschuhe |
| 1 | SMT | Ski-Handschuhe |
| 2 | HRO | Ski-Brille |

Ausschnitt aus der Tabelle *hersteller*:

| herstellercode | herstellername |
|---|---|
| NRG | Norgesta |
| HSK | Hasken |
| HRO | Herold |

### Join-Spalten in den Tabellen kunde und bundesland

Die Tabellen *kunde* und *bundesland* lassen sich durch die Spalte *bundesland* in der Tabelle *kunde* und *bcode* in der Tabelle *bundesland* verbinden. Wenn mehrere Kunden im selben Bundesland wohnen, tritt die Abkürzung für das Bundesland in mehreren Sätzen der Tabelle *kunde* auf.

Ausschnitt aus der Tabelle *kunde*:

| kunden_nr | vorname | nachname | ... | bundesland |
|---|---|---|---|---|
| 101 | Ludwig | Pauli | | BY |
| 102 | Karola | Sadler | | BY |
| 103 | Philipp | Korres | | BY |

Ausschnitt aus der Tabelle *bundesland*:

| bcode | laendername |
|---|---|
| BW | Baden-Wuerttemberg |
| BY | Bayern |
| BE | Berlin (West) |
| BR | Bremen |

## Daten in der Datenbank versand

Die Daten, die in der Datenbank *versand* gespeichert sind, finden Sie in den folgenden Tabellen.

### Tabelle kunde

| kunden_nr | vorname | nachname | firma | adresse1 | adresse2 | ort | bundesland | plz | telefon |
|---|---|---|---|---|---|---|---|---|---|
| 101 | Ludwig | Pauli | Pauli Sport | Forstweg 47 | | Augsburg | BY | 8900 | 0821/8075 |
| 102 | Karola | Sadler | Sport-Aktiv | Blumenstr. 12 | | Wasserburg | BY | 8090 | 08071/121289 |
| 103 | Philipp | Korres | Philipp's Sportwaren | Sandgasse 2 | Postfach 3498 | Rosenheim | BY | 8200 | 08031/4543 |
| 104 | Anton | Hochfeld | Spielball | Goethestr. 34 | Lilienstr. 42 | Muenchen | BY | 8000 | 089/25430 |
| 105 | Raimund | Viktor | Der Laden | Hofstr. 34 | | Ingolstadt | BY | 8070 | 0841/2321 |
| 106 | Georg | Watt | Sport Watt | Am Bergsteig 88 | | Ulm | BY | 7900 | 0731/7334421 |
| 107 | Karl | Remark | - Sportwaren - | Allee 13 | | Rosenheim | BY | 8200 | 08031/560324 |
| 108 | Martin | Korting | Martin's Shop | Buchenstr. 51 | | Muenchen | BY | 8000 | 089/33730 |
| 109 | Janette | Millet | Sportausstattung | Munsterstr. 1 | Wofgangstr 66d | Augsburg | BY | 8900 | 0821/358789 |
| 110 | Roland | Jaeger | Sportwaren | Herbststr. 77 | | Muenchen | BY | 8000 | 089/5032 |
| 111 | Frank | Keyser | Sport Keyser | Im Tal 2 | | Augsburg | BY | 8900 | 0821/7845 |
| 112 | Margarete | Larsinger | Larsinger & Partner | Wiesenweg 11 | | Ingolstadt | BY | 8070 | 0841/797235 |
| 113 | Liane | Bergen | Sporthaus | Hochstr. 59 | | Landshut | BY | 8300 | 0871/6699182 |
| 114 | Frank | Albert | Der Sport-Albert | Torstr. 33 | | Muenchen | BY | 8000 | 089/6367 |
| 115 | Alfred | Grantella | Sportladen | Forchenstr. 99 | | Landshut | BY | 8300 | 0871/498822 |
| 116 | Johannes | Partellman | Olympia Sport | Spinosastr. 10 | | Ulm | BY | 7900 | 0731/601123 |
| 117 | Arnold | Sipell | Sportecke | Marktplatz 4 | | Muenchen | BY | 8000 | 089/6321 |
| 118 | Dieter | Bachmann | Sportausstatter | Lindenweg 4 | | Passau | BY | 8390 | 0851/560011 |

## Tabelle auftrag

| auftrags_nr | auftragsdatum | kunden_nr | lieferhinweis | offen | fremd_nr | lieferdatum | liefergewicht | zustellgebuehr | zahldatum |
|---|---|---|---|---|---|---|---|---|---|
| 1001 | 20.01.1990 | 104 | Spedition | n | B77836 | 01.02.1990 | 20,40 | 10,00 | 22.03.1990 |
| 1002 | 01.06.1990 | 101 | Hintertuere klingeln | n | 9270 | 06.06.1990 | 50,60 | 15,30 | 03.07.1990 |
| 1003 | 12.10.1990 | 104 | Spedition | n | B77890 | 13.10.1990 | 35,60 | 10,80 | 04.11.1990 |
| 1004 | 12.04.1990 | 106 | zweimal klingeln | j | 8006 | 30.04.1990 | 95,80 | 19,20 | |
| 1005 | 04.12.1990 | 116 | vorher anrufen | n | 2865 | 19.12.1990 | 80,80 | 16,20 | 30.12.1990 |
| 1006 | 19.09.1990 | 112 | nach 10.00 Uhr | j | Q13557 | | 70,80 | 14,20 | |
| 1007 | 25.03.1990 | 117 | | n | 278693 | 23.04.1990 | 125,90 | 25,20 | |
| 1008 | 17.11.1990 | 110 | Samstags geschlossen | j | LZ230 | 06.12.1990 | 45,60 | 13,80 | 21.12.1990 |
| 1009 | 14.02.1990 | 111 | zweite Tuere rechts | n | 4745 | 04.03.1990 | 20,40 | 10,00 | 21.04.1990 |
| 1010 | 29.05.1990 | 115 | wenn geschlossen, nebenan abliefern | n | 429Q | 08.06.1990 | 40,60 | 12,30 | 22.07.1990 |
| 1011 | 23.03.1990 | 104 | Spedition | n | B77897 | 13.04.1990 | 10,40 | 5,00 | 01.06.1990 |
| 1012 | 05.06.1990 | 117 | | n | 278701 | 09.06.1990 | 70,80 | 14,20 | |
| 1013 | 01.09.1990 | 104 | Spedition | n | B77930 | 18.09.1990 | 60,80 | 12,20 | 10.10.1990 |
| 1014 | 01.05.1990 | 106 | mehrfach klingeln | n | 8052 | 10.05.1990 | 40,60 | 12,30 | 18.07.1990 |
| 1015 | 10.07.1990 | 110 | Samstags geschlossen | n | MA003 | 01.08.1990 | 20,60 | 6,30 | 31.08.1990 |

## Tabelle posten

| posten_nr | auftrags_nr | artikel_nr | herstellercode | menge | gesamtpreis |
|---|---|---|---|---|---|
| 1 | 1001 | 1 | HRO | 1 | 250,00 |
| 1 | 1002 | 4 | HSK | 1 | 960,00 |
| 2 | 1002 | 3 | HSK | 1 | 240,00 |
| 1 | 1003 | 9 | ANZ | 1 | 20,00 |
| 2 | 1003 | 8 | ANZ | 1 | 840,00 |
| 3 | 1003 | 5 | ANZ | 5 | 99,00 |
| 1 | 1004 | 1 | HRO | 1 | 960,00 |
| 2 | 1004 | 2 | HRO | 1 | 126,00 |
| 3 | 1004 | 3 | HSK | 1 | 240,00 |
| 4 | 1004 | 1 | HSK | 1 | 800,00 |
| 1 | 1005 | 5 | NRG | 10 | 280,00 |
| 2 | 1005 | 5 | ANZ | 10 | 198,00 |
| 3 | 1005 | 6 | SMT | 1 | 36,00 |
| 4 | 1005 | 6 | ANZ | 1 | 48,00 |
| 1 | 1006 | 5 | SMT | 5 | 125,00 |
| 2 | 1006 | 5 | NRG | 5 | 190,00 |
| 3 | 1006 | 5 | ANZ | 5 | 99,00 |
| 4 | 1006 | 6 | SMT | 1 | 36,00 |
| 5 | 1006 | 6 | ANZ | 1 | 48,00 |
| 1 | 1007 | 1 | HRO | 1 | 250,00 |
| 2 | 1007 | 2 | HRO | 1 | 126,00 |
| 3 | 1007 | 3 | HSK | 1 | 240,00 |
| 4 | 1007 | 4 | HRO | 1 | 480,00 |
| 5 | 1007 | 7 | HRO | 1 | 600,00 |
| 1 | 1008 | 8 | ANZ | 1 | 840,00 |
| 2 | 1008 | 9 | ANZ | 5 | 100,00 |
| 1 | 1009 | 1 | SMT | 1 | 450,00 |
| 1 | 1010 | 6 | SMT | 1 | 36,00 |
| 2 | 1010 | 6 | ANZ | 1 | 48,00 |
| 1 | 1011 | 5 | ANZ | 5 | 99,00 |
| 1 | 1012 | 8 | ANZ | 1 | 840,00 |
| 2 | 1012 | 9 | ANZ | 10 | 200,00 |
| 1 | 1013 | 5 | ANZ | 1 | 19,80 |
| 2 | 1013 | 6 | SMT | 1 | 36,00 |
| 3 | 1013 | 6 | ANZ | 1 | 48,00 |
| 4 | 1013 | 9 | ANZ | 2 | 40,00 |
| 1 | 1014 | 4 | HSK | 1 | 960,00 |
| 2 | 1014 | 4 | HRO | 1 | 480,00 |
| 1 | 1015 | 1 | SMT | 1 | 450,00 |

### Tabelle artikel

| artikel_nr | herstellercode | bezeichnung | preis | liefereinheit | stueckliste |
|---|---|---|---|---|---|
| 1 | HRO | Ski-Handschuhe | 250,00 | Box | 10/Box |
| 1 | HSK | Ski-Handschuhe | 800,00 | Box | 10/Box |
| 1 | SMT | Ski-Handschuhe | 450,00 | Box | 10/Box |
| 2 | HRO | Ski-Brille | 126,00 | Box | 24/Box |
| 3 | HSK | Ski-Stock | 240,00 | Col. | 12/Colli |
| 4 | HSK | Fussball | 960,00 | Col. | 24/Colli |
| 4 | HRO | Fussball | 480,00 | Col. | 24/Colli |
| 5 | NRG | Tennisschlaeger | 28,00 | solo | solo |
| 5 | SMT | Tennisschlaeger | 25,00 | solo | solo |
| 5 | ANZ | Tennisschlaeger | 19,80 | solo | solo |
| 6 | SMT | Tennisball | 36,00 | Box | 24 Dosen/Box |
| 6 | ANZ | Tennisball | 48,00 | Box | 24 Dosen/Box |
| 7 | HRO | Basketball | 600,00 | Box | 24/Box |
| 8 | ANZ | Volleyball | 840,00 | Box | 24/Box |
| 9 | ANZ | Volleyb. profi | 20,00 | solo | solo |

### Tabelle hersteller

| herstellercode | herstellername |
|---|---|
| SMT | Schmitt |
| ANZ | Anzabel |
| NRG | Norgesta |
| HSK | Hasken |
| HRO | Herold |

### Tabelle bundesland

| bcode | laendername |
|---|---|
| BW | Baden-Wuerttemberg |
| BY | Bayern |
| BE | Berlin (West) |
| BR | Bremen |
| HA | Hamburg |
| HE | Hessen |
| NI | Niedersachsen |
| NW | Nordrhein-Westfalen |
| RP | Rheinland-Pfalz |
| SA | Saarland |
| SH | Schleswig-Holstein |

# Formatprogramme

## auftrag

```
{ Datei: auftrag.per }

database
        versand

screen
{
--------------------------------------------------------------------------------
Kundennr: [k1          ]                    Telefon:  [k10           ]
   Firma: [k4                  ]
 Vorname: [k2              ]                Nachname:  [k3             ]
 Adresse: [k5                  ]                       [k6                  ]
     Plz: [k7   ] Ort: [k8             ] Bundesland: [k9]
--------------------------------------------------------------------------------
Auftragsnummer: [au11         ]        Auftragsdatum: [au12       ]
 Fremde Nummer: [au20       ]
         offen: [a]
   Lieferdatum: [au21       ]             Zahldatum: [au22       ]
--------------------------------------------------------------------------------
       Artikel: [p13]                          Preis: [preis ]
Herstellercode: [p16]                      x  Menge: [p18]
  Beschreibung: [bezeichnung     ]         ----------------------
       Einheit: [einh]                     =  Total: [p19       ]
    Hersteller: [hersteller      ]
   Stueckliste: [stueckliste     ] Auftr.-Gesamtpreis: [dtot        ]
}
end

tables
        kunde
        auftrag
        posten
        artikel
        hersteller

attributes

k1  = * kunde.kunden_nr = auftrag.kunden_nr, queryclear;
k2  = vorname, comments = "Bei Bedarf Vornamen eingeben";
k3  = nachname;
k4  = firma, reverse;
k5  = adresse1;
k6  = adresse2;
k7  = plz;
k8  = ort;
k9  = bundesland, upshift, autonext,
     include = ("BY","HA","BW","BE"), default = "BY",
     comments = "Erlaubte Bundeslaender sind BY, HA, BW, or BE";
k10 = telefon, picture = "#####/########";

au11 = * auftrag.auftrags_nr = posten.auftrags_nr, queryclear;
au12 = auftragsdatum, default = today;
au20 = fremd_nr;
```

```
a    = offen, downshift, default = "n", include = ("j","n");
au21 = lieferdatum;
au22 = zahldatum;

p13 = posten.artikel_nr, queryclear, include = (1 to 9),
      comments = "Artikelnummer zwischen 1 und 9";
    = * artikel.artikel_nr, noentry, noupdate, queryclear;
p16 = posten.herstellercode,
         lookup hersteller = hersteller.herstellername,
         joining * hersteller.herstellercode, upshift, queryclear,
         comments = "Codes sind HRO, HSK, SMT, NRG, SMT und ANZ";
    = * artikel.herstellercode, noentry, noupdate, queryclear;
p18 = menge, include = (1 to 100);
p19 = posten.gesamtpreis;

bezeichnung = artikel.bezeichnung, noentry, noupdate;
einh        = artikel.liefereinheit, noentry, noupdate;
stueckliste = artikel.stueckliste,  noentry, noupdate;
preis       = artikel.preis,  noentry, noupdate;

dtot        = displayonly type money;

instructions

kunde master of auftrag;
auftrag master of posten;

composites <posten.artikel_nr, posten.herstellercode>
          * <artikel.artikel_nr, artikel.herstellercode>

before editadd editupdate of posten.menge
        if preis is null
          then begin
            comments bell "Artikel und Hersteller passen nicht zusammen"
            nextfield = p13
        end

after editadd editupdate of posten.menge
        let p19  = preis * p18
        nextfield = p13

after add update query of posten
        let dtot = (total of p19)

end
```

## FORMAT01

```
database versand
screen
{
                                FORMAT01
                                --------

Kundennummer [f000       ]


Firma        [f001                ]


Vorname      [f002           ]       Nachname             [f003           ]
Strasse      [f004                ] evtl. Zweitanschrift [f005                ]
Plz          [f006 ]                 Ort                  [f007           ]


Telefon      [f008              ]    Bundesland           [a0]
}
end
tables
kunde
attributes
f000 = kunde.kunden_nr, reverse;
f001 = kunde.firma;
f002 = kunde.vorname;
f003 = kunde.nachname;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.plz;
f007 = kunde.ort;
f008 = kunde.telefon;
a0 = kunde.bundesland;
end
```

**FORMAT02**

```
database versand
screen
{

                        FORMAT02 - Bildschirmseite 1
                        ----------------------------


     Kundennummer       [f000       ]


     Firma              [f001                ]

     Vorname            [f002           ]
     Nachname           [f003           ]



}
screen
{
```

```
                        FORMAT02 - Bildschirmseite 2
                        ----------------------------


     Firma              [f001                ]


     Anschrift          [f004                ]
     evt. Zweitanschrift [f005               ]

     Plz                [f006 ]
     Ort                [f007          ]
     Bundesland         [a0]

     Telefon            [f008             ]

}
end
tables
kunde
attributes
f000 = kunde.kunden_nr, reverse;
f001 = kunde.firma;
f002 = kunde.vorname;
f003 = kunde.nachname;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.plz;
f007 = kunde.ort;
a0 = kunde.bundesland;
f008 = kunde.telefon;
end
```

## FORMAT03

```
database versand
screen
{
                    FORMAT03 - Tabellen 'kunde' und 'auftrag'
--kunde-------------------------------------------------------------------------

Firma     [f001                 ]
Vorname   [f002           ]           Nachname             [f003           ]
Adresse   [f004                ]      evtl. Zweitanschrift [f005                ]
Plz       [f006 ]                     Ort                  [f007           ]

Telefon   [f008              ]        Bundesland           [a0]

--auftrag------------ Kundennummer [f000       ] ------------------------------

Auftragsnummer      [f009       ]     Auftragsdatum      [f010      ]
Lieferhinweis       [f011                                      ]
Offen?              [a]               Fremdnummer        [f012      ]
Lieferdatum         [f013      ]      Liefergewicht      [f014      ]
Zustellgebuehr      [f015    ]        Zahldatum          [f016      ]
}
end
tables
kunde
auftrag
attributes
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr, reverse;
f001 = kunde.firma;
f002 = kunde.vorname;
f003 = kunde.nachname;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.plz;
f007 = kunde.ort;
a0   = kunde.bundesland;
f008 = kunde.telefon;

f009 = auftrag.auftrags_nr;
f010 = auftrag.auftragsdatum;
f011 = auftrag.lieferhinweis;
a    = auftrag.offen;
f012 = auftrag.fremd_nr;
f013 = auftrag.lieferdatum;
f014 = auftrag.liefergewicht;
f015 = auftrag.zustellgebuehr;
f016 = auftrag.zahldatum;
instructions
kunde master of auftrag
end
```

## FORMAT04

```
database versand
screen
{

--kunde-------------------------------------------------------------------------

Firma     [f001                ]   Adresse          [f004                ]
Plz       [f006 ]                  Ort              [f007           ]
                                   Bundesland       [a ]
--auftrag----------- Kundennummer [f000       ] --------------------------------

Offen?              [b]            Auftragsdatum    [f010       ]
Lieferdatum         [f013       ]  Liefergewicht    [f014       ]
Zustellgebuehr      [f015     ]    Zahldatum        [f016       ]

--posten---------- Auftragsnummer [f009       ] --------------------------------

Postennummer        [f017  ]       Herstellercode   [c ]
Artikelnummer       [f018  ]       Artikel          [lo01                ]
Menge               [f019  ]       Gesamtpreis      [f020       ]
}
end
tables
    kunde
    auftrag
    posten
    artikel
attributes
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr, reverse;
f001 = kunde.firma,
       comments = "Bei Neuaufnahme: falls 'GmbH', bitte mit eingeben!";
f004 = kunde.adresse1, required;
f006 = kunde.plz;
f007 = kunde.ort;
a    = kunde.bundesland, upshift, default="BY"
f009 = * auftrag.auftrags_nr
     =   posten.auftrags_nr, reverse;
b    = auftrag.offen,
       include = (j, n),
       comments = "'j' oder 'n' eingeben";
f010 = auftrag.auftragsdatum, default = today;
f013 = auftrag.lieferdatum, verify;
f014 = auftrag.liefergewicht;
f015 = auftrag.zustellgebuehr;
f016 = auftrag.zahldatum;
f017 = posten.posten_nr;
c    = posten.herstellercode;
f018 = posten.artikel_nr,
       lookup lo01 = artikel.bezeichnung
       joining artikel.artikel_nr;
f019 = posten.menge, right,
       include = (1 to 50),
       comments = "Mindestens 1, maximal 50";
f020 = posten.gesamtpreis;
instructions
kunde master of auftrag;
auftrag master of posten;
end
```

## FORMAT05

```
database versand
screen
{
 \gp-------------------------------------------------------------q\g
 \g|\g                      FORMAT05                             \g|\g
 \g|\g                      --------                             \g|\g
 \g|-------------------------------------------------------------|\g
 \g|\g                                                           \g|\g
 \g|\g                                                           \g|\g
 \g|\g                                                           \g|\g
 \g|\g   Kundennummer       [f000       ]                       \g|\g
 \g|\g                                                           \g|\g
 \g|\g                                                           \g|\g
 \g|\g   Firma              [f001                ]              \g|\g
 \g|\g                                                           \g|\g
 \g|\g   Vorname            [f002           ]                   \g|\g
 \g|\g   Nachname           [f003           ]                   \g|\g
 \g|\g                                                           \g|\g
 \gb-------------------------------------------------------------d\g


}
end
tables
k=kunde
attributes
f000 = k.kunden_nr, reverse;
f001 = k.firma;
f002 = k.vorname;
f003 = k.nachname;
end
```

## muster

```
{ Datei: muster.per }
database versand
screen
{
================================================================================
================================================================================
KUNDENDATEN:

   Kundennummer: [k1          ]

          Firma: [k4                    ]
        Vorname: [k2         ]              Nachname: [k3              ]

        Adresse: [k5                    ]
                 [k6                    ]

            Plz: [k7  ]   Ort: [k8              ]     Bundesland: [k9]
        Telefon: [k10           ]


================================================================================
================================================================================
}
screen
{
================================================================================
KUNDENNUMMER: [k1          ]           FIRMA: [k4                    ]
AUFTRAGSDATEN:
 Auftragsnummer: [au11         ]    Auftragsdatum: [au12          ]
  Artikelnummer: [p13         ]       Hersteller: [p16]
   Beschreibung: [ar14           ]                 [m17            ]
    Stueckliste: [ar16           ]
                                              Menge: [p18         ]
                                              Preis: [ar15              ]
                                         Gesamtpreis: [p19              ]

LIEFERHINWEIS:
  Fremde Nummer: [au20       ]      Liefergebuehr: [d1         ]

          offen: [a]          Auftragsgesamtsumme: [d2         ]
    Lieferdatum: [au21       ]
      Zahldatum: [au22       ]
  Lieferhinweis: [au23                                      ]
}
end
tables
kunde posten artikel
auftrag hersteller
attributes

k1   = * kunde.kunden_nr
         = auftrag.kunden_nr;
k2   = vorname,
       comments = "Bei Bedarf Vornamen eingeben";
k3   = nachname;
```

```
k4   = firma, reverse;
k5   = adresse1;
k6   = adresse2;
k7   = plz, autonext;
k8   = ort;
k9   = bundesland, upshift, autonext,
       include = ("HH","BY","BW","BE"),
       default = "BY" ,
       comments = "Eingabe erlaubt:'HH','BY','BW','BE'- Standardwert: 'BY'";
k10  = telefon, picture = "#####/########";
au11 = * auftrag.auftrags_nr
         = posten.auftrags_nr;
au12 = auftragsdatum, default = today, format = "dd.mm.yy";
p13  = posten.artikel_nr;
       = * artikel.artikel_nr, noentry, noupdate, queryclear;
p16  =  posten.herstellercode,
           lookup m17 = hersteller.herstellername
           joining *hersteller.herstellercode, upshift, autonext;
        = * artikel.herstellercode, noentry, noupdate,
            upshift, autonext, queryclear;
ar14 = artikel.bezeichnung, noentry, noupdate;
ar15 = artikel.preis, noentry, noupdate;
ar16 = artikel.stueckliste, noentry, noupdate;
p18  = posten.menge, include = (1 to 50),
       comments = "Menge zwischen 1 und 50" ;
p19  = posten.gesamtpreis;
au20 = fremd_nr, required,
       comments = "Falls keine fremde Nummer, Name des Anrufers" ;
a    = offen, autonext;
au21 = lieferdatum, default = today, format = "dd.mm.yy";
au22 = zahldatum, format = "dd.mm.yy";
au23 = lieferhinweis;
d1   = displayonly type money;
d2   = displayonly type money;

instructions
kunde master of auftrag;
auftrag master of posten;
composites <posten.artikel_nr, posten.herstellercode>
           *<artikel.artikel_nr, artikel.herstellercode>
before editadd editupdate of auftrag
   nextfield = au20
before editadd editupdate of posten
   nextfield = p13
after editadd editupdate of menge
   let p19 = p18 * ar15
   nextfield = au11
after add update query of posten
   if (total of p19) <= 100 then
      let d1 = 7.50
      else
      let d1 = (total of p19) * .04
   let d2 = (total of p19) + d1
after display of auftrag
   let d1 = 0
   let d2 = 0
end
```

# Listenprogramme

## auftrag1

```
{ Datei: auftrag1.ace - Quellprogramm fuer 1. Auftragsliste }

database
   versand
end

output
   page length 72                  {Standardlaenge von 'lpr'      }
   report to "aufliste1"
end

select
   auftrag.auftrags_nr aufnr,
   auftragsdatum, kunden_nr,
   fremd_nr, lieferdatum, zustellgebuehr,
   zahldatum,
   posten.auftrags_nr, artikel_nr, herstellercode,
   menge, gesamtpreis
   from auftrag, posten
   where auftrag.auftrags_nr = posten.auftrags_nr
   order by aufnr
end

format

   before group of aufnr
   print "Auftragsnummer:   ", aufnr using "#####",
      " fuer Kundennummer:   ", kunden_nr using "#####"
   print 3 spaces, "Fremdnummer:   ", fremd_nr,
      "Auftragsdatum:   ", auftragsdatum
   skip 1 line
   print "Art.nr", column 20,
      "Herst.", column 28, "Mge", column 42, "Preis"

   on every row
   print artikel_nr using "###", column 20,
      herstellercode, column 28, menge using "###",
      column 38, gesamtpreis using "$$$,$$$.&&"
   after group of aufnr
   skip 1 line
   print 5 spaces, "Gesamtpreis des Auftrags:     ",
      group total of gesamtpreis using "$$,$$$,$$$.&&"
   skip 3 lines
end
```

**auftrag2**

```
{ Datei: auftrag2.ace - Quellprogramm fuer 2. Auftragsliste }

database
   versand
end

define
   variable startdatum date
   variable enddatum date
end

input
   prompt for startdatum using
      "Bitte geben Sie das Startdatum fuer die Liste ein:  "
   prompt for enddatum using
      "Bitte geben Sie das Enddatum fuer die Liste ein:    "
end

output
   left margin 0
   page length 72                {Standardlaenge von 'lpr'     }
   report to "aufliste3"
end

select
   kunde.kunden_nr, vorname, nachname, firma,

   auftrag.auftrags_nr aufnr, auftrag.kunden_nr,
      auftragsdatum, month(auftragsdatum) months,
      day(auftragsdatum) tage, year(auftragsdatum) jahre,

   posten.auftrags_nr, menge, gesamtpreis

   from  kunde, auftrag, posten

   where kunde.kunden_nr = auftrag.kunden_nr
      and auftrag.auftrags_nr = posten.auftrags_nr and
      auftragsdatum between $startdatum and $enddatum

   order by jahre, months, tage, firma, aufnr
end
```

```
format

   first page header
   print column 10,

      "=================================================="
   print column 10,
      "                    TAGEWEISE AUFTRAGSLISTE"
   print column 10,
      "=================================================="
   skip 1 line
   print column 15, "VOM:    ", startdatum
      using "dd/mm/yy",
      column 35, "BIS ZUM: ", enddatum
      using "dd/mm/yy"
   print column 15, "Liste vom:  ",
      today using "dd mmm, yyyy"
   skip 2 lines
   print column 2, "AUFTR. VOM", column 15,
      "FIRMA", column 35, "NAME",
      column 55, "NUMMER", column 66, "UMSATZ"

   before group of tage
   skip 2 lines

   after group of aufnr
   print column 2, auftragsdatum, column 15,
      firma clipped, column 35, vorname clipped,
      1 space, nachname clipped, column 55,
      aufnr using "####", column 60,
      group total of gesamtpreis using "$$,$$$,$$$.&&"

   after group of tage
   skip 1 line
   print column 21, "Auftragssumme fuer den Tag:  ",
      group total of gesamtpreis using "$$$$,$$$,$$$.&&"
   skip 1 line
   print column 15,
      "=================================================="

   on last row
   skip 1 line
   print column 15,
      "=================================================="
   skip 2 lines
   print "Gesamtauftragssumme:  ", total of
      gesamtpreis using "$$$$,$$$,$$$.&&"

   page trailer
   print column 28, pageno using "Seite <<<<"
end
```

## kliste1

```
{ Datei: kliste1.ace - Quellprogramm fuer 1. Kundenliste }

database
   versand
end

output
   left margin 2
end

select
   kunden_nr,
   vorname,
   nachname,
   firma,
   ort,
   bundesland,
   plz,
   telefon

   from kunde

   order by ort
end

format
   first page header
   print column 33, "KUNDENLISTE"
   print column 33, "-----------"
   skip 2 lines
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
   skip 1 line

   page header
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
   skip 1 line

   on every row
   print kunden_nr using "####",
      column 9, vorname clipped, 1 space, nachname clipped,
      column 32, plz,
      column 40, ort clipped, ", " , bundesland,
      column 62, telefon

   on last row
   skip 1 line
   print  "ANZAHL DER KUNDEN:",
      column 30, count using "##"
end
```

### kliste2

```
{ Datei: kliste2.ace - Quellprogramm fuer 2. Kundenliste }

database
   versand
end
define
   variable diesland char(2)
end
input
   prompt for diesland using
   "Bitte Bundesland der Kunden angeben (in GROSSBUCHSTABEN):  "
end
output
   left margin 0
end
select
   kunden_nr,
   vorname,
   nachname,
   firma,
   ort,
   bundesland,
   plz,
   telefon
   from kunde
   where bundesland matches $diesland
   order by plz, nachname
end
format
   first page header
   print column 33, "KUNDENLISTE"
   print column 33, "-----------"
   skip 2 lines
   print "Liste fuer Bundesland ", diesland
   skip 2 lines
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
   skip 1 line
   page header
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
```

```
    skip 1 line
    on every row
    print kunden_nr using "####",
       column 9, vorname clipped, 1 space, nachname clipped,
       column 32, plz,
       column 40, ort clipped, ", " , bundesland,
       column 62, telefon
    on last row
    skip 2 lines
    print "Anzahl der Kunden in ",diesland, " ist ",
       count using "<<<<&"
end
```

## LISTE01

```
{ Datei: LISTE01.ace }

database versand
end

define variable firmenname char(20)
end

input prompt for firmenname
using "Geben Sie den Namen der Firma ein: "
end

output report to printer
       page length 72          {Standardlaenge von 'lpr'}
end

select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
where
   firma = $firmenname
end

format

page header
print "K u n d e n d a t e n:"
skip 2 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
end
```

## LISTE02

```
{ Datei: LISTE02.ace }

database versand
end

output report to printer
       page length 72           {Standardlaenge von 'lpr'}
end

select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
end

format

page header
print column 30, "Kunden-Liste"
skip 1 line
print "Stand: ", today
print "-------------------------------------------------------------------"
skip 2 line

page trailer
print "-------------------------------------------------------------------"
print "Kunden-Liste", column 54, "Seite: ", pageno using "##"

on every row
need 8 lines
print "K u n d e n d a t e n:"
skip 1 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
skip 2 line
end
```

## LISTE03

```
{ Datei: LISTE03.ace }

database versand
end

output report to printer
        page length 72          {Standardlaenge von 'lpr'}
end

select
   nachname, firma, ort
from
   kunde
order by
   ort
end

format

page header
print column 18, "Kunden-Liste"
print column 18, "============"
skip 2 lines

before group of ort
print "STADT: ", ort
skip 1 line

on every row
print nachname, column 18, firma, column 40, ort
after group of ort
skip 2 lines
end
```

**LISTE04**

```
{ Datei: LISTE04.ace }

database versand
end

define variable prozent decimal(3,2)
end

input prompt for prozent
      using "Die Artikel sollen erhoeht werden um Prozent: "
end

output report to printer
        page length 72          {Standardlaenge von 'lpr'}
end

select
   herstellercode, bezeichnung, preis
from
   artikel
end

format

page header
print "PROZENTSATZ: ", prozent using "##.##"
let prozent = prozent / 100 + 1
skip 1 line
print      "HERSTELLER",
column 12, "BEZEICHNUNG",
column 32, "PREIS",
column 43, "PLUS",
column 49, "NEUPREIS"
skip 1 line

on every row
print herstellercode,
column 12, bezeichnung clipped,
column 30, preis using "####.##",
column 40, preis * prozent - preis using "####.##",
column 50, preis * prozent using "####.##"
end
```

## LISTE05

```
{ Datei: LISTE05.ace }

database versand end

output report to printer
        page length 72          {Standardlaenge von 'lpr'}
end

select
   kunde.kunden_nr knr, firma,
   auftrag.auftrags_nr anr
from
   kunde, auftrag
where
   kunde.kunden_nr = auftrag.kunden_nr
order by
   knr
end

format

first page header
print            "KUNDENNUMMER",
      column 20, "FIRMA",
      column 37, "AUFTRAGSNUMMER"
skip 2 lines

on every row
print column 10, knr using "###",
      column 20, firma clipped,
      column 47, anr using "####"

after group of knr
print "-------------------------------------------------"
print column 30, "ANZAHL AUFTRAEGE ",
      column 49,  group count using "##"
print "-------------------------------------------------"

on last row
skip 2 lines
print "================================================="
print column 23, "GESAMT-ANZAHL AUFTRAEGE ",
      column 49 , count using "##"
print "================================================="
end
```

**post1**

```
{ Datei: post1.ace - Quellprogramm fuer 1. Anschriftenliste

Dieses Listenprogramm gibt eine Anschriftenliste aus. Diese ist
nach den Postleitzahlen und den Nachnamen geordnet. Die
PRINT-Anweisung im FORMAT-Abschnitt erlaubt es, mehrere Felder in
einer Zeile auszugeben.
Wenn es fuer ein Feld in einer Anschrift keine Daten gibt, dann
stehen an der entsprechenden Stelle Leerzeichen.

Zwischen den Feldern einer Anschrift sind mehr Leerzeichen als
noetig. Im Listen-Quellprogramm 'post2.ace' werden solche
Leerzeichen abgeschnitten.
}

database
   versand
end

select *
   from kunde
   order by plz, nachname
end

format
   on every row
   print vorname, nachname
   print firma
   print adresse1
   print adresse2
   print plz, 2 spaces, ort, ", " , bundesland
   skip 2 lines
end
```

## post2

```
{ Datei: post2.ace - Quellprogramm fuer 2. Anschriftenliste

Dieses Listenprogramm gibt wie 'post1.ace' eine Anschriftenliste
aus. Es ist im Vergleich zu diesem erweitert:

-  Die Raender und die Seitenlaenge werden abweichend vom
   Standard gesetzt und die Ausgabe erfolgt in eine
   Datei (OUTPUT-Abschnitt).

-  Im FORMAT-Abschnitt gibt es IF-Anweisungen und ueberfluessige
   Leerzeichen am Ende von Feldern werden abgeschnitten.
}

database
   versand
end
output
   top margin 0
   bottom margin 0
   left margin 0
   page length 9
   report to "anschriften2"
end
select
   vorname, nachname,
   firma,
   adresse1,
   adresse2,
   ort, bundesland, plz
   from kunde
   order by plz, nachname
end
format
   on every row
   if (ort is not null) and
      (bundesland is not null) then
   begin
      print vorname clipped, 1 space, nachname
      print firma
      print adresse1
      if (adresse2 is not null) then
         print adresse2
      print plz, 2 spaces, ort clipped,
         ", " , bundesland
      skip to top of page
   end
end
```

**post3**

```
{ Datei: post3.ace - Quellprogramm fuer 3. Anschriftenliste

Dieses Listenprogramm schreibt eine Anschriftenliste in einer,
zwei oder in drei Spalten in eine Datei. Es speichert jeweils
bis zu drei Anschriften in drei Zeichenketten; wenn diese
entsprechend gefuellt sind, gibt das Programm sie aus. Nach dem
Aufruf des Programms, legt der Benutzer die Anzahl der Spalten
fest.
}

database
   versand
end

define
   variable name     char(75)     {fuer Vor- und Nachnamen      }
   variable plobl    char(75)     {fuer PLZ, Ort, B.land        }
   variable z_kette1 char(80)     {Zeichenkette fuer 1. Zeile   }
   variable z_kette2 char(80)     {Zeichenkette fuer 2. Zeile   }
   variable z_kette3 char(80)     {Zeichenkette fuer 3. Zeile   }
   variable start    smallint     {Start- und Endposition einer }
   variable ende     smallint     {Adresse in Zeichenkette      }
   variable a_breite smallint     {Anschriftenbreite            }
   variable platz    smallint     {Anschriftenzwischenraum      }
   variable sp_anz   smallint     {Anzahl der Listenspalten     }
   variable i        smallint     {Zaehler fuer Listenspalte    }
end

input
   prompt for sp_anz using
   "Wieviel Spalten soll die Liste haben? [1-3] "
end

output
   top margin 0
   bottom margin 0
   left margin 0
   page length 72                 {Standardlaenge von 'lpr'     }
   report to "anschriften3"
end

select *
   from kunde

   order by plz
end

format
```

```
   first page header                {In diesem Abschnitt wird
                                     nichts ausgegeben. Das Pro-
                                     gramm versieht nur Variable
                                     mit Werten.                 }
   let i = 1                        {i initialisieren           }
   let a_breite = 72/sp_anz         {legt die Anschriftenbreite
                                     fest                        }
   let platz = 8/sp_anz             {legt den Platz zwischen zwei
                                     Anschriftenspalten fest     }

   on every row
   let name = vorname clipped, 1 space, nachname
   let plobl = plz, 2 spaces, ort clipped,
               ", ", bundesland

   let ende = (i * a_breite)        {Aktuelle Endposition be-   }
              + platz               {rechnen                    }
   let start = ende - a_breite      {Aktuelle Startposition be- }
                                    {rechnen                    }
   let z_kette1[start, ende]        {Namen an richtige Stelle in}
      = name                        {die Zeichenkette schreiben }
   let z_kette2[start, ende]        {Str. an richtige Stelle in }
      = adresse1                    {die Zeichenkette schreiben }
   let z_kette3[start, ende]        {Ort an richtige Stelle in  }
      = plobl                       {die Zeichenkette schreiben }
   if i = sp_anz then               {Falls alle Spalten in den  }
   begin                            {Zeichenketten, schreibe    }
      print z_kette1 clipped        {die Namen,                 }
      print z_kette2 clipped        {die Strassen und           }
      print z_kette3 clipped        {die Orte.                  }
      skip 1 line
      let z_kette1 = " "            {Zeichenketten mit Leerzeichen}
      let z_kette2 = " "            {belegen                    }
      let z_kette3 = " "
      let i  = 1                    {Aktuellen Spaltenzaehler fuer}
   end                              {Anschriften zuruecksetzen  }
   else
      let i = i + 1

   on last row
   if i > 1 then                    {Die uebriggebliebenen An-  }
   begin                            {schriften werden ausgegeben}
      print z_kette1 clipped
      print z_kette2 clipped
      print z_kette3 clipped
   end
end
```

## A.2 Dienstprogramme

Folgende Dienstprogramme, die auf Betriebssystemebene aufzurufen sind, stehen dem INFORMIX-Anwender bzw. INFORMIX-Datenbankverwalter zur Verfügung:

| Dienstprogramm | Bedeutung | gültig für Backend |
|---|---|---|
| *bcheck* | Indexprüfprogramm | SE |
| *dbexport* | Datenbank exportieren | SE, ON |
| *dbimport* | Datenbank importieren | SE, ON |
| *dbload* | Daten aus Datei in Datenbank einlesen | SE,ON |
| *dblog* | Transaktionsprotokoll-Dateien ausgeben | SE |
| *dbschema* | Anweisungen für Datenbankaufbau erzeugen | SE, ON |
| *tbcheck* | Indexprüfprogramm | ON 1) |
| *tbload* | Datenbank oder Tabelle binär laden | ON 1) |
| *tblog* | logische Protokolle ausgeben | ON 1) |
| *tbunload* | Datenbank oder Tabelle binär entladen | ON 1) |

SE = INFORMIX-SE
ON = INFORMIX-ONLINE

1) Die Beschreibung dieses Dienstprogramms finden Sie im ONLINE-Handbuch [4].

### A.2.1 bcheck - Indexprüfprogramm

*bcheck* ist ein Dienstprogramm, das Indexdateien prüft oder auch repariert. Es kann nur für INFORMIX-SE-Datenbanken verwendet werden. INFORMIX-ONLINE-Anwender benutzen anstelle von *bcheck* das Dienstprogramm *tbcheck* (siehe INFORMIX-ONLINE-Handbuch [4]). *bcheck* vergleicht die Indexdatei mit der Datendatei. Geprüft wird die Übereinstimmung von Spalten, die als Index vereinbart wurden, mit den in der Indexdatei geschriebenen Schlüsselwerten. Wenn es keine Übereinstimmung gibt, fragt Sie das Prüfprogramm, ob Sie den defekten Index löschen und neu aufbauen wollen. Wenn Sie erneuern lassen wollen, liest *bcheck* die ganze Datendatei sequentiell durch und schreibt die gefundenen Indexwerte wieder in die Indexdatei. Voraussetzung ist natürlich, daß die Indexbeschreibung in der Indexdatei noch in Ordnung ist.

```
bcheck[␣-ilnjqs]␣datei[...]
```

| | |
|---|---|
| **-i** | Nur die Indexdatei wird geprüft. |
| **-l** | Alle Einträge in den b-trees werden aufgelistet. |
| **-n** | Alle Fragen, die gestellt werden, werden mit nein beantwortet. |
| **-j** | Alle Fragen, die gestellt werden, werden mit ja beantwortet. |
| **-q** | Ohne Meldungen. |
| **-s** | Verändert die Knotengröße in der Indexdatei. |

datei Name der zu prüfenden C-ISAM-Datei.

Wenn Sie keinen Parameter angeben, wartet *bcheck* auf eine Antwort von Ihnen, wenn ein Fehler gefunden wurde. Verwenden Sie den Parameter **-j** immer mit Vorsicht und erst ab der 2. Überprüfung der Datei.

### Beispiel für eine normal verlaufende bcheck-Sitzung

```
$ bcheck kunde__100


BCHECK  C-ISAM B-tree Pruefprogramm   Version 4.00.U
Copyright (C) 1981-1989 Informix Software, Inc.


C-ISAM Datei: kunde__100

Pruefen der Dateibeschreibung und Datei-Groessen.
Aktuelle Knotengroesse der C_ISAM Index-Datei = 1024
Pruefen der Datendatei Saetze
Pruefen der Indizes- und Schluesselbeschreibungen.
Index 1 = Schluessel Eindeutig:
    0 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen
Index 2 = Schluessel Eindeutig:  (0,4,2)
    1 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen
Index 3 = Schluessel Duplikate:  (111,5,0)
    1 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen
Pruefen der Freiplatzlisten fuer Datensaetze und Index-Knoten
4 benutzte Index-Knoten, 0 frei — 18 benutzte Datensaetze, 0 frei
```

*bcheck* fand keine Fehler. Die Zahlen, die für jeden Index ausgegeben wurden, bedeuten die Stellung des Schlüsselwertes im Datensatz. Es können bis zu 8 Zahlengruppen für jeden Index angegeben werden.

### Beispiele für fehlerhafte Indexdateien

Im folgenden Fall soll *bcheck* keine Änderungen vornehmen. Beantworten Sie alle Fragen mit "nein", indem Sie das Programm mit -n aufrufen:

```
$ bcheck -n kunde__100


BCHECK  C-ISAM B-tree Pruefprogramm   Version 4.00.U
Copyright (C) 1981-1989 Informix Software, Inc.


C-ISAM Datei: kunde__100

Pruefen der Dateibeschreibung und Datei-Groessen.
Knotengroesse der Index-Datei = 1024
Aktuelle Knotengroesse der C_ISAM Index-Datei = 1024
Pruefen der Datendatei Saetze
Pruefen der Indizes- und Schluesselbeschreibungen.
Index 1 = Schluessel Eindeutig:
    0 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen
Index 2 = Schluessel Eindeutig:  (0,4,2)
    1 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen

FEHLER: 1 fehlende Zeiger auf Datensaetze
Index entfernen ? nein

Index 3 = Schluessel Duplikate:  (111,5,0)
    1 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen

FEHLER: 1 fehlende Zeiger auf Datensaetze
Index entfernen ? nein

Pruefen der Freiplatzlisten fuer Datensaetze und Index-Knoten

FEHLER: 1 mehrfache Zeiger auf Datensaetze
Freiplatzliste fuer Datensaetze wiederherstellen ?nein

4 benutzte Index-Knoten, 0 frei — 19 benutzte Datensaetze, 0 frei
```

Im folgenden Fall wollen Sie die Indizes löschen und neu erstellen. Beantworten Sie diesmal alle Fragen mit "ja", indem Sie das Programm mit -j aufrufen:

```
$ bcheck -j kunde__100
```

```
BCHECK  C-ISAM B-tree Pruefprogramm   Version 4.00.U
Copyright (C) 1981-1989 Informix Software, Inc.


C-ISAM Datei: kunde__100

Pruefen der Dateibeschreibung und Datei-Groessen.
Knotengroesse der Index-Datei = 1024
Aktuelle Knotengroesse der C_ISAM Index-Datei = 1024
Pruefen der Datendatei Saetze
Pruefen der Indizes- und Schluesselbeschreibungen.
Index 1 = Schluessel Eindeutig:
    0 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen
Index 2 = Schluessel Eindeutig:  (0,4,2)
    1 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen

FEHLER: 3 fehlende Zeiger auf Datensaetze
Index entfernen ? ja

Index wiederherstellen ? ja

Index 3 = Schluessel Duplikate:  (111,5,0)
    1 benutzte Index-Knoten — 1 benutzte Index b-tree Ebenen

FEHLER: 3 fehlende Zeiger auf Datensaetze
Index entfernen ? ja

Index wiederherstellen ? ja

Pruefen der Freiplatzlisten fuer Datensaetze und Index-Knoten

FEHLER: 3 mehrfache Zeiger auf Datensaetze
Freiplatzliste fuer Datensaetze wiederherstellen ?ja

Freiplatzliste fuer Datensaetze wird wiederhergestellt.
Index 3 wird wiederhergestellt.
Index 2 wird wiederhergestellt.

FEHLER: Mehrfach vorhandener Schluesselwert, Datensatz 2

FEHLER: Mehrfach vorhandener Schluesselwert, Datensatz 12

FEHLER: Mehrfach vorhandener Schluesselwert, Datensatz 13
4 benutzte Index-Knoten, 0 frei — 21 benutzte Datensaetze, 0 frei
```

### A.2.2 dbexport - Datenbank exportieren

Mit diesem Dienstprogramm erzeugen Sie aus einer Datenbank ASCII-Dateien, die Sie mit dem Dienstprogramm *dbimport* in eine andere Datenbankumgebung importieren können.

*dbexport* kann nur vom Datenbankverwalter (Zugriffsrecht DBA) oder dem Benutzer *informix* aufgerufen werden.

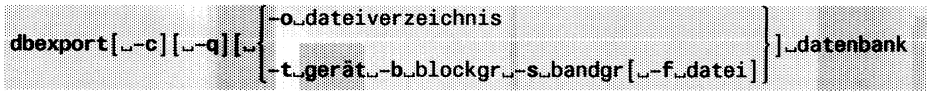
*Ausgabe auf Platte: dbexport* erzeugt im angegebenen Dateiverzeichnis (Schalter **-o**) das Dateiverzeichnis *datenbank.exp*, das folgende Dateien enthält:

| | |
|---|---|
| `tabelle1.` *unl*<br>...<br>...<br>...<br>`tabellen.` *unl* | Je Tabelle eine Datei, die die Daten enthält. |
| datenbank.*sql* | Datei, die die SQL-Anweisungen zum Datenbankaufbau enthält. |
| *dbexport.out* | Datei, die das Ablaufprotokoll enthält. |

*datenbank.exp* wird der Gruppe *informix* zugeordnet.

*Ausgabe auf Band: dbexport* erzeugt kein Dateiverzeichnis. Die Dateien werden direkt auf Band ausgegeben.

Stellen Sie vor dem *dbexport*-Aufruf sicher, daß die Umgebungsvariable SQLEXEC auf die gewünschte Umgebung (ONLINE oder SE gesetzt ist).



**-c**

gibt an, daß das Programm fortgesetzt werden soll, wenn Fehler auftauchen, ausgenommen folgende schwerwiegende Fehler:

- Das angegebene Bandgerät kann nicht eröffnet werden.
- Fehlerhafte Schreibzugriffe auf Band oder Platte.
- Ungültige Parameter im Kommando.
- Datenbank kann nicht eröffnet werden oder keine Zugriffsberechtigung.

**-q**

unterdrückt evtl. vorhandene Meldungen von SQL-Anweisungen am Bildschirm.

**-o** dateiverzeichnis

Exportiert die Datenbank auf Platte.
Geben Sie den absoluten oder relativen Pfadnamen eines Dateiverzeichnisses an, in dem die von *dbexport* erzeugten Dateien abgelegt werden sollen. Fehlt die Angabe, so gilt das aktuelle Dateiverzeichnis.

**-t** gerät

Exportiert die Datenbank auf Band.
Geben Sie die Gerätedatei des Bandgerätes an.
Wenn die Datenbank auf ein Band exportiert wird, dürfen Sie *dbexport* nicht als Hintergrundprozeß aufrufen.

**-b** blockgr

Blockgröße des Bandes in Kbyte.

Für INFORMIX-ONLINE gilt: Die Blockgröße muß ein Vielfaches der Page-Größe sein.

**-s** bandgr

Kapazität des Bandes in Kbyte. Die Größe, die Sie hier angeben, sollte mindestens 10% unter der tatsächlichen Größe des Bandes liegen. Sie könnten sonst beim Sichern Schwierigkeiten bekommen.

**-f** datei

schreibt die von *dbexport* erzeugten SQL-Anweisungen nicht auf Band, sondern in die angegebene Datei auf Platte. Fehlt die Angabe, so werden
die SQL-Anweisungen in die Datei datenbank.**sql** auf Band ausgegeben.

datenbank
Name der Datenbank, die exportiert werden soll.

**Hinweis**

- Während des Programmablaufs ist die Datenbank exklusiv gesperrt. Falls dies nicht möglich ist, erscheint eine Fehlermeldung.
- Mit der Taste [DEL] können Sie den Programmablauf abbrechen. Dabei werden Sie aufgefordert, dies nochmals zu bestätigen.
- Die Datenbank wird als "Datenbank ohne Transaktionssicherung" exportiert. Soll die Datenbank beim Import (*dbimport*) wieder mit Transaktionssicherung arbeiten, so müssen Sie dies bei *dbimport* entsprechend angeben
(Schalter **-l**).

**Beispiel**

Das folgende Kommando exportiert die Datenbank *versand* auf Band. Die Blockgröße des Bandes beträgt 16 Kbytes; jedes Band soll eine Datenmenge von 40000 Kbytes aufnehmen. Bei Fehlern soll das Programm fortfahren.

```
dbexport -c -t /dev/rts -b 16 -s 40000 versand
```

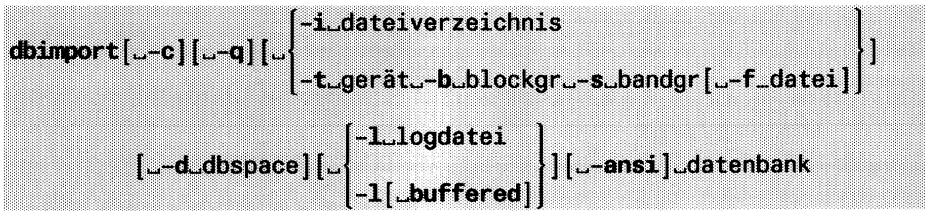
Das folgende Kommando exportiert die Datenbank *buecher* in das Dateiverzeichnis */usr/lomata/port*:

```
dbexport -c -o /usr/lomata/port buecher
```

### A.2.3 dbimport - Datenbank importieren

Mit diesem Dienstprogramm erzeugen Sie eine Datenbank aus den Dateien, die Sie zuvor mit *dbexport* erzeugt haben. Die Kennung, unter der *dbimport* abläuft, erhält das DBA-Zugriffsrecht auf die Datenbank.

Stellen Sie vor dem *dbimport*-Aufruf sicher, daß die Umgebungsvariable SQLEXEC auf die gewünschte Umgebung (ONLINE oder SE gesetzt ist).




**-c** gibt an, daß das Programm fortgesetzt werden soll, wenn Fehler auftauchen, ausgenommen folgende schwerwiegende Fehler:

- Das angegebene Bandgerät kann nicht eröffnet werden.
- Fehlerhafte Lesezugriffe auf Band oder Platte.
- Ungültige Parameter im Kommando.
- Datenbank kann nicht eröffnet werden oder keine Zugriffsberechtigung.

**-q** unterdrückt Meldungen von SQL-Anweisungen am Bildschirm.

**-i** dateiverzeichnis

Importiert die Datenbank von Platte.
Geben Sie den absoluten oder relativen Pfadnamen eines Dateiverzeichnisses an, in dem die von *dbexport* erzeugten Dateien enthalten sind. Fehlt die Angabe, so gilt das aktuelle Dateiverzeichnis, d. h. *dbimport* sucht das Dateiverzeichnis datenbank**.exp** im aktuellen Dateiverzeichnis.

**-t** gerät
Importiert die Datenbank vom Band.
Geben Sie die Gerätedatei des Bandgerätes an.
Wenn die Datenbank vom Band importiert wird, dürfen Sie *dbimport* nicht als Hintergrundprozeß aufrufen.

**-b** blockgr
Blockgröße des Bandes in Kbyte. Die Angabe muß mit der Blockgröße übereinstimmen, die Sie bei *dbexport* angegeben haben.

**-s** bandgr
Kapazität des Bandes in Kbyte. Die Angabe muß mit der Bandgröße übereinstimmen, die Sie bei *dbexport* angegeben haben.

**-f** datei
Name der Datei, die auch bei *dbexport* (Schalter **-f**) angegeben ist.

**-d** dbspace
Nur für INFORMIX-ONLINE:
Name des Dbspaces, in dem die Datenbank eingerichtet werden soll. Fehlt die Angabe, so wird die Datenbank im Root-Dbspace eingerichtet.

**-l**
schaltet für die importierte Datenbank die Transaktionssicherung ein. Fehlt die Angabe, so wird keine Transaktionssicherung durchgeführt. Der Schalter **-l** muß angegeben werden, wenn die neue Datenbank als ANSI-Datenbank erzeugt wird (Angabe **-ansi**).

logdatei
Nur für INFORMIX-SE:
Absoluter Pfadname der Datei, in die das Transaktionsprotokoll geschrieben werden soll.

**buffered**
Nur für INFORMIX-ONLINE:
wählt gepufferte Protokollierung. Diese Angabe ist bei ANSII-Datenbanken verboten. Fehlt die Angabe, so gilt als Standardwert: ungepufferte Protokollierung.

**-ansi**
erzeugt die neue Datenbank als ANSI-Datenbank (MODE ANSI).
Wenn Sie diese Angabe verwenden, so müssen Sie auch den Schalter **-l** angeben.

datenbank

Bei Import von Platte gilt: Name der Datenbank, wie er bei *dbexport* angegeben war. Die importierte Datenbank erhält denselben Namen.

Bei Import von Band gilt: Name, den die importierte Datenbank erhalten soll. Er muß nicht mit dem bei *dbexport* angegebenen Datenbanknamen übereinstimmen.

**Hinweis**

- Während des Programmablaufs ist die Datenbank exklusiv gesperrt.
- Beim Laden der Datenbank führt *dbimport* einen impliziten LOCK TABLE und UNLOCK TABLE durch.
- INFORMIX-SE: Die Datenbankdateien der importierten Datenbank werden im aktuellen Dateiverzeichnis angelegt.
- Ist der Schalter -l nicht angegeben, d. h. die Transaktionssicherung nicht eingeschaltet, so können Sie dies nachträglich durchführen:

  INFORMIX-ONLINE: über den DB-Monitor
  (siehe ONLINE-Handbuch [4])

  INFORMIX-SE: über die SQL-Anweisung START DATABASE
  (siehe SQL-Handbuch [1])
- Mit der Taste [DEL] können Sie den Programmablauf abbrechen. Dabei werden Sie aufgefordert, dies nochmals zu bestätigen.
- *dbimport* erzeugt die Meldungsdatei *dbimport.out.* Diese enthält alle Fehlermeldungen und Warnungen, die während des Programmablaufes entstehen.
- INFORMIX-ONLINE: Extent-Größe festlegen

  Die Extent-Größe für Tabellen stellt einen wichtigen Faktor für die Performance dar. Der Standardwert für die Größe des Initial-Extent von importierten Tabellen ist die Größe der importierten Tabelle. Der Standardwert für die nachfolgenden Extents beträgt 10 % der Größe des Initial-Extent. Falls zu erwarten ist, daß die importierte Tabelle wesentlich größer werden wird, können Sie diese Standardwerte durch Angabe eigener Größen außer Kraft setzen (siehe Beispiel im ONLINE-Handbuch [4]).

Dazu editieren Sie die Anweisungsdatei datenbank.**sql** im **.exp**-Dateiverzeichnis, die *dbexport* beim Exportieren erzeugt hat oder - falls Schalter -f verwendet wurde - die dort angegebene Datei. Die Größen für die Initial-Extents und die nachfolgenden Extents können Sie in der CREATE TABLE-Anweisung hinzufügen (Beschreibung siehe SQL-Handbuch [1]).
Danach kann *dbimport* mit der geänderten Anweisungsdatei datenbank.**sql** gestartet werden.

**Beispiel**

Eine Datenbank auf Band (Gerät /dev/rts soll als INFORMIX-ONLINE-Datenbank in den Dbspace *dbspace2* importiert werden. Die Datenbank soll den Namen *versand* erhalten. Bildschirmmeldungen durch SQL-Anweisungen sollen unterdrückt werden. Bei Fehlern soll das Programm fortgesetzt werden:

```
dbimport -c -t /dev/rts -b 16 -s 40000 -q -d dbspace2 versand
```

Die Datenbank *buecher* befindet sich auf Platte und soll als INFORMIX-ONLINE-Datenbank in den Root-Dbspace importiert werden. Das Dateiverzeichnis *versand.exp* befindet sich im Dateiverzeichnis */usr/lomata/port*:

```
dbimport -i /usr/lomata/port buecher
```

Beachten Sie, daß für den Import in die INFORMIX-ONLINE-Umgebung die Umgebungsvariable SQLEXEC entsprechend gesetzt sein muß.

### A.2.4 dbload - Daten aus Datei in Datenbank einlesen

*dbload* ist ein Dienstprogramm, mit dem Sie Daten im ASCII-Format aus einer Betriebssystemdatei in eine Datenbank übertragen können. Es bietet Ihnen einen einfachen und leistungsfähigen Transfer von Daten aus einer anderen INFORMIX-Datenbank oder einem völlig anderem Datenbanksystem in Ihre Datenbank.

Die wichtigsten Funktionen von *dbload* sind:

- Daten aus bestimmten Feldern in einer oder mehreren Dateien können in ausgewählte Spalten einer oder mehrerer Datenbank-Tabellen geladen werden.
- Das Laden kann ab einer beliebig definierbaren Zeile der Eingabedatei beginnen.
- Beim Ladevorgang wird jeweils eine bestimmte von Ihnen definierbare Anzahl an Sätzen gemeinsam bearbeitet.
- Die Dateien können Sätze mit fester oder variabler Länge enthalten.
- Für jedes Feld kann ein bestimmter Wert als NULL-Wert definiert werden. Enthält ein Satz in diesem Feld den definierten Wert, so wird anstelle dieses Wertes ein NULL-Wert eingetragen.
- In der Anweisungsdatei können Sie auch direkt Werte angeben.
- Sätze, die nicht geladen werden können, werden zusammen mit einer Diagnose-Information in einer Fehlerdatei protokolliert.
- Sie können ein Fehlerlimit definieren, d. h. angeben, wieviel fehlerhafte Sätze während des Ladevorgangs maximal auftreten dürfen, ohne daß der Ladevorgang abgebrochen wird.
- Für Datenbanken mit Transaktionen können Sie festlegen, ob bei Abbruch des Ladevorgangs wegen Überschreitung des Fehlerlimits alle seit letztem COMMIT erfolgreich eingelesenen Sätze zurückgesetzt werden sollen.

#### Vorgehensweise

Für den *dbload*-Aufruf benötigen Sie die *Eingabedateien*, die die einzulesenden Daten enthalten, sowie eine *Anweisungsdatei*, die das Format der Eingabesätze beschreibt und entsprechend den Tabellenspalten zuordnet.

Die folgenden Abschnitte beschreiben den Aufbau von Eingabe- und Anweisungsdatei. Im Anschluß daran finden Sie die Syntax des *dbload*-Aufrufs.

### Aufbau der Eingabedatei

Analog zur Aufteilung von Tabellensätzen in einzelne Spalten müssen die Sätze der Eingabedatei in einzelne Felder unterteilbar sein. Ein Feld des Eingabesatzes läßt sich somit genau einer Tabellenspalte zuordnen. Welche Bereiche des Eingabesatzes jeweils zu einem Feld zusammengefaßt werden, definieren Sie in der Anweisungsdatei.

Weitere Merkmale der Eingabedatei:

- Sie darf nur abdruckbare Zeichen enthalten.
- Die einzelnen Sätze müssen mit dem Zeilenende-Zeichen abgeschlossen sein. Soll das Zeilenendezeichen als Feldinhalt verwendet werden, so muß es mit Gegenschrägstrich \ entwertet werden.
- Die Satzlänge kann entweder fest oder variabel sein:
  - *Feste Satzlänge* bedeutet, daß alle Sätze gleich lang sind und daß die Feldpositionen der sich entsprechenden Inhalte in jedem Satz übereinstimmen müssen. Enthält z. B. ein Eingabesatz ab Spalte 9 als Feldinhalt einen Nachnamen, so müssen sinngemäß in allen Eingabesätzen die Nachnamen ab Spalte 9 enthalten sein und auch dieselbe Länge haben (ggf. mit Leerzeichen auffüllen).
    In der Anweisungsdatei können Sie für Sätze fester Länge einen bestimmten Feldinhalt als NULL-Wert definieren. Beim Ladevorgang wird dann anstelle des dieses Feldinhaltes ein NULL-Wert übertragen.
  - *Variable Satzlänge* bedeutet, daß die sich entsprechenden Felder jedes Satzes (und damit jeder Satz) unterschiedliche Länge haben. Die Feldgrenzen müssen daher durch ein bestimmtes Zeichen markiert sein. Welches Zeichen als Feldtrennzeichen interpretiert werden soll, legen Sie in der Anweisungsdatei fest. Soll das Feldtrennzeichen als Feldinhalt interpretiert werden, so muß es mit Gegenschrägstrich \ entwertet werden.
    Zwei aufeinanderfolgeden Feldtrennzeichen bezeichnen ein leeres Feld. Beim Ladevorgang wird dafür ein NULL-Wert übertragen.

- Für alphanumerische Feldinhalte gilt: Stimmt die Länge nicht überein mit der Länge der Tabellenspalte, so wird beim Ladevorgang entprechend gekürzt oder mit Leerzeichen aufgefüllt.
- Die Feldinhalte müssen den Datentypen der Tabellenspalten entsprechen, denen Sie später zugeordnet werden:
  - Enthält ein Feldinhalt führende Leerzeichen, so darf er nur Tabellenspalten zugeordnet werden, für die der Datentyp DATE, MONEY oder ein numerischer Datentyp definiert ist. Felder, die Tabellenspalten vom Datentyp MONEY zugeordnet werden, dürfen zusätzlich das Währungssymbol enthalten. Es muß an erster Stelle stehen.
  - Felder, die Tabellenspalten vom Datentyp DATE zugeordnet werden, müssen das Datum in dem Format enthalten, das in der Umgebungsvariablen DBDATE definiert ist.
  - Felder, die Tabellenspalten vom Datentyp DATETIME oder INTERVAL zugeordnet werden, müssen die einzelnen Zeitangaben in der Form yyyy-mm-dd-hh:mm:ss enthalten (siehe Beschreibung dieser Datentypen im SQL-Handbuch [1].
  - INFORMIX-ONLINE: Felder, die Tabellenspalten vom Datentyp BYTE zugeordnet sind, müssen Daten im ASCII-Format enthalten. Führende Leerzeichen sind nicht erlaubt.

Ein einfaches Beispiel für eine zulässige Eingabedatei stellt die von der SQL-Anweisung UNLOAD erzeugte Datei dar. Diese enthält Sätze variabler Länge. Die einzelnen Felder durch ein bei UNLOAD definiertes Feldtrennzeichen gekennzeichnet (UNLOAD-Beschreibung siehe SQL-Handbuch [1]).

### Aufbau der Anweisungsdatei

Die Anweisungsdatei ordnet die Daten der Eingabedateien den Tabellen der Datenbank zu. Sie enthält zweierlei Arten von Anweisungen:

- FILE-Anweisung: Dort geben Sie den Namen einer Eingabedatei an und definieren die Felder der Eingabesätze.
- INSERT-Anweisung: Dort ordnen Sie die in der FILE-Anweisung definierten Felder den Tabellenspalten zu.

Die Anweisungsdatei enthält immer eine Folge von FILE- und INSERT-Anweisungen, wobei für die Anordnung gilt: Auf eine FILE-Anweisung folgen immer eine oder mehrere INSERT-Anweisungen, die sich auf diese FILE-Anweisung beziehen. In einer Anweisungsdatei lassen sich mehrere solcher FILE-INSERT-Folgen definieren.

### Syntax der Anweisungen

Kommentare innerhalb der Anweisungsdatei sind in geschweifte Klammern { } zu setzen. Mit Ausnahme der letzten Anweisung müssen sämtliche Anweisungen mit Semikolon ; abgeschlossen werden.

Die erste FILE-Anweisung zeigt die Syntax für Sätze variabler Länge, die zweite die für Sätze fester Länge.

```
FILE␣"datei"␣DELIMITER␣"c"␣feldanzahl;      {für variable Satzlänge}

FILE␣"datei"                                    {für feste Satzlänge}

     (feld1␣start[-end][:␣...] [␣NULL = "str1"],
      feld2␣start[-end][:␣...] [␣NULL = "str2"],
      ...
      ...
      feldn␣start[-end][:␣...] [␣NULL = "strn"]);

INSERT␣INTO␣tabelle [(spaltenname,...)] [VALUES␣(wert,...)]; [...;]
```

FILE
: definiert die Eingabedatei und die Eingabefelder. Je Eingabedatei muß 1 FILE-Anweisung definiert werden.

"datei"
: Name der Eingabedatei, eingeschlossen in Anführungszeichen "". Die Benutzerkennung, unter der *dbload* abläuft, muß Leseberechtigung auf die Datei haben.

DELIMITER
: bezeichnet das Feldtrennzeichen für Sätze variabler Länge. Das Feldtrennzeichen muß innerhalb der gesamten Eingabedatei einheitlich sein.
*dbload* ordnet automatisch jedem Feld des Eingabesatzes die Feldnamen **f01, f02, f03, ...** usw. zu. Diese Feldnamen müssen Sie in der INSERT-Anweisung angeben, wenn Sie auf die Felder Bezug nehmen.

"c"
: Feldtrennzeichen, eingeschlossen in Anführungszeichen "". Das Feldtrennzeichen muß auch am Ende des letzten Feldes (vor dem Zeilenendezeichen) gesetzt werden. Wenn Sie das Feldtrennzeichen am Ende weglassen, läuft *dbload* auf Fehler, falls das letzte Feld leer ist, weil dann die Feldanzahl nicht mehr stimmt.

feldanzahl
: Anzahl Felder im Eingabesatz; die Anzahl muß in allen Sätzen übereinstimmen.

feld1 ... feldn
: hier definieren Sie Feldnamen für Eingabesätze fester Länge. Die nachfolgenden INSERT-Anweisungen, die sich auf diese FILE-Anweisung beziehen, verwenden diese Feldnamen.

start-end
: bezeichnet den Wertebereich im Eingabesatz, der unter dem definierten Feldnamen angesprochen werden soll. *start* bezeichnet die Anfangsspalte, *end* die Endspalte des Wertes im Eingabesatz. Sie können hier für den angegebenen Feldnamen ein Feld aus mehreren Wertebereichen zusammensetzen; Sie müssen dann die einzelnen Wertebereiche *start* - *end* durch den Doppelpunkt trennen (z. B. 1-10 : 15-18 usw.). Dabei dürfen sich die angegebenen Wertebereiche auch überlappen, z. B. 1-10 : 5-22.
  Fehlt die Angabe von *end*, so gilt als Standardwert: 1 Zeichen ab *start*.

NULL="str1"..."strn"
: definiert den Feldinhalt, der als NULL-Wert interpretiert werden soll, d. h. anstelle dieses Feldinhalts wird beim Laden der NULL-Wert übertragen. Für jedes Feld läßt sich ein eigene Zeichenfolge definieren, die als NULL-Wert zu interpretieren ist.

INSERT INTO
: hier definieren Sie eine INSERT-Anweisung. Die Angabe des Operanden *select-anweisung* ist nicht erlaubt. Ansonsten entspricht die Syntax der von INSERT (siehe SQL-Handbuch [1]).

tabelle
: bezeichnet die Tabelle, in die die Daten geladen werden sollen. Die Benutzerkennung, unter der *dbload* läuft, muß INSERT-Zugriffsrecht für die angegebene Tabelle haben.
  Der Tabellenname darf auch in der Form *eigentümer.tabelle* angegeben werden.

(spaltenname,...)
: bezeichnet die Spalten der Tabelle, in die Werte einzutragen sind. Ausführliche Beschreibung siehe INSERT-Anweisung im SQL-Handbuch [1]. Für Tabellenspalten, die hier nicht angegeben sind, versucht INFORMIX NULL-Werte zu übertragen. Falls dies aufgrund der Tabellendefinition nicht möglich ist, erscheint eine Fehlermeldung bei Ablauf des Programms.
  Fehlt die Angabe, so gelten alle Spalten der Tabelle in der dort definierten Reihenfolge.

VALUES
: bezeichnet die Liste der Werte, die den angegebenen Spalten zuzuordnen sind.

(wert,...)

Mit *wert* bezeichnen Sie entweder direkt eine Konstante, die in die Spalte übertragen werden soll oder Sie geben den Feldnamen an, wie er in der vorangehenden FILE-Anweisung definiert wurde.
Für die Zuordnung der Werte gilt: die Werte der Werteliste werden in der definierten Reihenfolge den Spalten der angegebenen Spaltenliste zugeordnet (1. Wert der Wertelist wird 1. Spalte der Spaltenliste zugeordnet usw.).
Die Anzahl der Werte in der Werteliste muß übereinstimmen mit der Anzahl der Spalten in der Spaltenliste.

### Syntax des dbload-Aufrufs

*Vor dem Aufruf beachten*

Fehlt eine der Angaben *-d datenbank, -c anweisungsdatei* oder *-l fehlerprotokoll*, so fragt *dbload* unabhängig vom Standardwert sämtliche im Aufruf fehlenden Operanden im Dialog ab. Soll für einen Operanden der Standardwert angenommen werden, so müssen Sie bei Abfrage dieses Operanden eine "leere" Eingabe machen, d. h. die Taste ↵ drücken.

Enthält der *dbload*-Aufruf alle drei oben genannten Operanden, so setzt *dbload* für alle anderen nicht angegebenen Operanden automatisch den definierten Standardwert ein.

```
dbload[␣-d[␣datenbank]][␣-c[␣anweisungsdatei]][␣-l[␣fehlerprotokoll]]
      [␣-e[␣anz1]][␣-n[␣anz2]][␣-i[␣anz3]][␣-p][␣-r][␣-s [ ausgabedatei]]
```

**-d** datenbank

Name der Datenbank, in die die Daten übertragen werden sollen. Fehlt hier der Datenbankname, so wird er von *dbload* automatisch angefordert.

**-c** anweisungsdatei
Name der Anweisungsdatei, die die erforderlichen Anweisungen enthält. Fehlt hier der Name der Anweisungsdatei, so wird er von *dbload* automatisch angefordert.

**-l** fehlerprotokoll
Name für das Fehlerprotokoll. Fehlt hier der Name des Fehlerprotokolls, so wird er von *dbload* automatisch angefordert.

**-e** anz1
bezeichnet die Anzahl der fehlerhaften Sätze, die bearbeitet werden, ehe *dbload* abbricht. Standardwert: 10 Das heißt: *dbload* bricht beim elften fehlerhaften Satz ab.

**-n** anz2
bezeichnet die Anzahl der Sätze, die von *dbload* erfolgreich gelesen werden müssen, ehe sie in die Datenbank übertragen werden. Bevor diese Anzahl nicht erreicht ist, werden keine Sätze in die Datenbank übertragen. Standardwert: 100 Das heißt: Jeweils Einheiten zu 100 Sätzen werden übertragen. *dbload* gibt nach jeder übertragen Einheit eine Meldung aus.

**-i** anz3
bezeichnet die Anzahl der Sätze in der Datei, die beim Einlesen übersprungen werden sollen. Das heißt: Der erste eingelesene Satz ist der Satz, der auf die angegebene Anzahl Sätze folgt.
Diese Angabe ist vor allem nützlich,

- wenn Sie einen zuvor abgebrochenen Ladevorgang an der unterbrochenen Stelle fortsetzen wollen.
- wenn Sie allgemeine Kommentare am Anfang der Eingabedatei überspringen wollen.

**-p**
Nur für Datenbanken mit Transaktionssicherung:
Bewirkt, daß bei Abbruch von *dbload* eine Abfrage erscheint, ob die bisher eingelesenen (aber nicht übertragenen) Sätze übertragen (≙ COMMIT) oder zurückgesetzt (≙ ROLLBACK) werden sollen. Standardwert: Die Sätze werden übertragen.

**-r**
weist *dbload* an, keine Tabellen zu sperren.

**-s** prüft nur die Syntax der Anweisungen in der Anweisungsdatei. Es werden keine Daten in die Datenbank übertragen. Diese Angabe verwenden Sie vor dem eigentlichen Laden der Daten. Die Anweisungsdatei wird zeilenweise auf dem Bildschirm ausgegeben. Dabei werden Syntaxfehler gekennzeichnet.

> ausgabedatei
schreibt das Ergebnis der Syntaxprüfung in die angegebene Datei.

**Hinweis**

- Sämtliche Dateinamen können auch mit dem Pfadnamen angegeben werden.
- Für Datenbanken mit Transaktionssicherung gilt: Wenn Sie *dbload* selbst abbrechen (Taste DEL), dann werden alle seit dem letzten COMMIT eingelesenen Sätze wieder zurückgesetzt.
- Die Geschwindigkeit, in der die Daten übertragen werden, kann beeinträchtigt werden, wenn die Empfänger-Tabelle Indizes hat. Daher empfiehlt es sich, die Indizes vor dem *dbload*-Aufruf zu löschen und danach neu zu erzeugen.

**Beispiel für den dbload-Aufruf**

*dbload*-Aufruf, der mit Ausnahme von **-s** sämtliche Operanden zeigt:

```
dbload -d db -c adat -l fdat -e 5 -n 75 -i 20 -p -r
```

Die Angaben bedeuten:

| | |
|---|---|
| -d db | Die Sätze sollen in die Datenbank *db* übertragen werden. |
| -c adat | Die Anweisungsdatei heißt *adat.* |
| -l fdat | Die Datei für das Fehlerprotokoll heißt *fdat.* |
| -e 5 | *dbload* bricht beim sechsten fehlehaften Satz ab. |
| -e 75 | *dbload* überträgt die Sätze jeweils in Einheiten zu 75 fehlerfreien Sätzen. |
| -i 20 | *dbload* ignoriert die ersten 20 Zeilen der Eingabedatei. |
| -p | bei Abbruch von *dbload* erscheint eine Abfrage, was mit bereits eingelesenen, aber noch nicht übertragenen Sätzen geschehen soll. |
| -r | die Tabellen, in die Daten übertragen werden, sollen während des Ladevorgangs nicht gesperrt werden. |

### Beispiel für den Aufbau einer Anweisungsdatei

```
FILE datei1 (feld1  1 - 10 : 13      :  5 - 22  NULL = "string1",
             feld2 10 - 21 : 28 - 32,
             feld3  8 - 10 : 33 - 50 : 29 - 33  NULL = "string2",
             ...
             ...
             feldn  9      : 16 - 19            NULL = "stringn");

INSERT INTO tab1 (col1, col2, col9, ... , coln) ;

INSERT INTO tab2 VALUES (feld1, feld3, "abc", ... , feldn) ;

INSERT INTO tab3 ;      {ohne Angabe von Feldern und Werten }

FILE "datei.2" DELIMITER "|" num       {variable Feldlänge}

INSERT INTO tab1 VALUES (f01, f02, "abc", "234", ... , f0n) ;

INSERT INTO tab4 ;
```

Bedeutung der einzelnen Anweisungen:

```
-  FILE datei1 (feld1  1 - 10 : 13      :  5 - 22  NULL = "string1",
               feld2 10 - 21 : 28 - 32,
               feld3  8 - 10 : 33 - 50 : 29 - 33  NULL = "string2",
               ...
               ...
               feldn  9      : 16 - 19            NULL = "stringn");
```

| | |
|---|---|
| datei1 | ist der Name der Eingabedatei. |
| feld1 ... feldn | Feldnamen, die Datenfelder fester Länge identifizieren. |

In dem gezeigten Beispiel setzt sich *feld1* zusammen aus den Zeichen 1 bis 10, Zeichen 13 und Zeichen 5 bis 22 eines jeden Satzes der Datei.

Zeichenpositionen dürfen beliebig wiederholt werden. Im Beispiel erscheinen die Zeichenpositionen 5 bis 10 und 13 zweimal in *feld1*; die Zeichenpositionen 10, 13 und 21 erscheinen sowohl in *feld1* als auch in *feld2.*

In *feld1* ist der NULL-Wert als "string1" definiert. *dbload* setzt in das Feld immer dann einen NULL-Wert, wenn es auf "string1" trifft.

*feld2* besteht aus den Zeichen in Position 10 bis 21 und 28 bis 32.

*feld3* besteht aus den Zeichen in Position 8 bis 10, 33 bis 50 und 29 bis 33. Der NULL-Wert in diesem Feld ist als "string2" definiert.

Die Felddefinition wird fortgesetzt, bis das letzte Feld erreicht ist. Hier enthält *feldn* die Zeichen in Postionen 9 und 16 bis 19. Der NULL-Wert in diesem Feld ist als "stringn" definiert.

Die Felddefinitionen sind in runde Klammern () einzuschließen. Zeichen und Zeichenfolgen sind durch einen Doppelpunkt : zu trennen.

- `INSERT INTO tab1 (col1, col2, col9, ... , coln) ;`

  Es folgt eine INSERT-Anweisung. Da es keine Werteliste gibt, nimmt *dbload* die Werteliste der vorhergehenden FILE-Anweisung. *dbload* überträgt die Daten von *feld1* nach *col1*, die von *feld2* nach *col2* und die von *feld3* nach *col9* der Tabelle *tab1*. (In diesem Fall werden die Tabellenspalten 4 bis 8 übergangen). Dieser Vorgang wird fortgesetzt, bis die Daten von *feldn* in das Feld *coln* übertragen sind.

- `INSERT INTO tab2 VALUES (feld1, feld3, "abc", ... , feldn) ;`

  Da es keine Spaltenliste gibt, gelten aus *tab2* alle Spalten, und zwar in der Reihenfolge, in der sie in den Systemtabellen abgspeichert sind. Die Werte, die in jedes Feld geladen werden sollen, werden eigens angegeben. *dbload* überträgt die Daten von *feld1* in die 1. Spalte der Tabelle, die von *feld3* in die 2. Spalte und die Konstante "abc" in die 3. Spalte der Tabelle *tab2*. Dieser Vorgang wird fortgesetzt, bis der Wert von *feldn* in das letzte Feld übertragen ist.

- `INSERT INTO tab3 ;` `{ohne Angabe von Feldern und Werten }`

  Da weder Feld- noch Werteliste angegeben ist, gelten alle Felder von *tab3* in der abgespeicherten Reihenfolge und die Werteliste aus der vorhergehenden FILE-Anweisung.

  Hier wird der Wert von *feld1* in die 1. Spalte übertragen, der von *feld2* in die 2. Spalte usw.

*Anmerkung:* Diese Anweisung erfordert es, daß die Feldliste in der FILE-Anweisung eine genaue (1:1) Entsprechung zu den Feldern der Tabelle enthält. Ist diese Entsprechung nicht gegeben, lädt *dbload* die Sätze nicht. Sie erhalten stattdessen für jeden Satz eine Fehlermeldung. *dbload* beendet den Ladevorgang, wenn die Gesamtzahl der fehlerhaften Sätze die vom Benutzer definierte Grenze (falls angegeben) oder den Standardwert von 10 fehlerhaften Sätzen überschreitet.

- `FILE "datei.2" DELIMITER "|" num {variable Feldlänge}`

  Die Felder in *datei.2* haben variable Länge. Mit der DELIMITER-Klausel wird als Feldtrennzeichen ein senkrechter Strich "|" definiert. Dieses Feldtrennzeichen muß in der gesamten Datei verwendet werden und ist in Anführungszeichen einzuschließen. 'num' bezeichnet die Anzahl der Felder, die einen Satz in der Datendatei bilden. Den Feldern werden automatisch die Namen *f01, f02,f03* usw. zugewiesen. Jedem Feld, das kein Zeichen enthält, wird ein NULL-Wert zugewiesen.

  Der Kommentar "{variable Feldlänge}" ist in geschweiften Klammern eingeschlossen.

- `INSERT INTO tab1 VALUES (f01, f02, "abc", "234", ... , f0n) ;`

  Da es keine Spaltenliste gibt, gelten alle Tabellenspalten von *tab1* in der abgespeicherten Reihenfolge. Werte, die in jede Spalte von *tab1* übertragen werden sollen, werden eigens angegeben. Hier wird der Wert von *f01* in die 1. Spalte übertragen, der von *f02* in die 2. Spalte, die Konstante "abc" in die 3. Spalte, der Wert "234" in die 4. Spalte usw.

- `INSERT INTO tab4 ;`

  Da weder Spalten- noch Werteliste angegeben ist, gelten alle Spalten der Tabelle *tab4* in der abgespeicherten Reihenfolge. Die Werteliste wird aus der vorhergehenden FILE-Anweisung genommen. Die Feldliste in der FILE-Anweisung muß eine genaue (1:1) Entsprechung zu den Spalten der Tabelle *tab4* enthalten. Hier wird der Wert von *f01* in die 1. Spalte übertragen, der von *f02* in die 2. Spalte usw.

### A.2.5 dblog - Transaktionsprotokoll-Dateien ausgeben

Das Dienstprogramm *dblog* gibt den Inhalt von Transaktionsprotokoll-Dateien aus, die mit INFORMIX-SE oder C-ISAM erstellt wurden.

Dieses Dienstprogramm steht nur dem Anwender von INFORMIX-SE zur Verfügung, INFORMIX-ONLINE-Anwender benutzen anstelle von *dblog* das Dienstprogramm *tblog* (siehe Handbuch INFORMIX-ONLINE).

```
dblog[␣-v][␣-l][␣-h␣anzahl]␣protokolldatei
      [␣-d␣anfangsdatum␣endedatum]
      [␣-t␣anfangszeit␣endezeit]
      [␣-u␣benutzer]
      [␣-f␣dateiname]
      [␣-r␣anfangsposition␣endeposition]
```

**-v** zusätzliche Information für spezielle Typen von Protokollsätzen ausgeben.

**-l** Information über Postition der Protokollsätze in der Protokolldatei ausgeben.

**-h** anzahl
gibt an, nach wieviel Zeilen die Überschrift bei der Ausgabe wiederholt werden soll.
Wenn Sie 0 angeben, wird die Überschrift nur einmal am Anfang ausgegeben.
Standard ist 20.

protokolldatei
Name der Transaktionsprotokoll-Datei.

**-d** anfangsdatum endedatum
Es werden nur die Aktivitäten von Anfangsdatum bis Endedatum ausgegeben. Das Format ist mm/tt; mm = Monat, tt = Tag

**-t** anfangszeit endezeit
Es werden nur die Aktivitäten von Anfangszeit bis Endezeit ausgegeben.

**-u** benutzer
Es werden nur die Aktivitäten der angegebenen Benutzerkennung ausgegeben.

**-f** dateiname
Es werden nur die Aktivitäten ausgegeben, die sich auf die Datenbank-Tabelle beziehen, die in *dateiname* gespeichert ist.

**-r** anfangsposition endeposition
Es werden nur die Aktivitäten von Anfangs- bis Endeposition der Protokollsätze ausgegeben. Die Position eines Protokollsatzes wird in Anzahl Bytes vom Anfang der Protokolldatei angegeben.

### Ausgabe der Protokollsätze

Für jeden Protokollsatz werden standardmäßig folgende Felder ausgegeben:

| | |
|---|---|
| TY | Typ des Protokollsatzes (siehe Ausgabe bei -v) |
| TrxID | Transaktionsnummer |
| User | SINIX-Benutzernummer, die zum Protokollsatz gehört. |
| User Name | Benutzerkennung, die zur Benutzernummer gehört. |
| Date/Time | Datum und Zeit, zu der der Protokollsatz geschrieben wurde. |
| Lngth | Länge des Protokollsatzes. |
| FD | Dateizeiger (File-Deskripotor) der betroffenen C-ISAM-Datei. |
| Recno | Rowid des Datensatzes, den der Protokollsatz betrifft. |
| Filename | Dateiname der vom Protokollsatz betroffenen Datei; für Satztypen open, close, build, erase und rename. |

**Ausgabe bei -l**

Wenn Sie mit dem Schalter **-l** die Ausgabe der Position der Protokollsätze anfordern, werden zusätzlich folgende drei Felder ausgegeben:

| | |
|---|---|
| Location | Position des Protokollsatzes; die Position wird in Anzahl Bytes vom Dateianfang der Protokolldatei ausgegeben. |
| Prev Loc | Position des vorhergehenden Protokollsatzes in derselben Transaktion wie der aktuelle Protokollsatz (nicht bei allen Protokollsatz-Typen verwendet) |
| Prev Len | Länge des vorhergehenden Protokollsatzes in derselben Transaktion wie der aktuelle Protokollsatz (nicht bei allen Protokollsatz-Typen verwendet) |

**Ausgabe bei -v**

Wenn Sie mit dem Schalter **-v** Zusatzausgaben anfordern, werden für einige Protokollsatz-Typen zusätzliche Felder ausgegeben. Die folgende Übersicht zeigt die Protokollsatz-Typen und die zusätzlich ausgegebenen Felder.

| **Protokollsatz** | **Code** | **zusätzlich ausgegebene Felder** |
|---|---|---|
| ISAM-Datei aufbauen | BU | Row Length (ISAM-Satzlänge)<br>Build Mode (C-ISAM-Modus der ISAM-Datei) |
| ISAM-Datei schließen | FC | keine zusätzlichen Felder |
| ISAM-Datei öffenen | FO | keine zusätzlichen Felder |
| Datensatz aus ISAM-Datei löschen | DE | Image (Hexadezimal und ASCII-Ausgabe des Protokollsatzes) |
| Datensatz in ISAM-Datei einfügen | IN | Image (Hexadezimal- und ASCII-Ausgabe des Protokollsatzes) |
| ISAM-Datei umbenennen | RE | Neuer Dateiname, der im Protokollsatz enthalten ist |

| | | |
|---|---|---|
| Datensatz in ISAM-Datei ändern | UP | Pre-Image (Hexadezimal- und ASCII-Ausgabe des Before Image des Datensatzes im Protokollsatz)<br>Post-Image (Hexadezimal- und ASCIIAusgabe des Aftere Image des Datensatzes im Protokollsatz) |
| Unique Id für ISAM-Datei anfordern | UN | keine zusätzlichen Felder |
| Unique Id für ISAM-Datei setzen | SU | keine zusätzlichen Felder |
| ISAM-Datei löschen | ER | keine zusätzlichen Felder |
| BEGIN WORK | BW | keine zusätzlichen Felder |
| COMMIT WORK | CW | keine zusätzlichen Felder |
| ROLLBACK WORK | RW | keine zusätzlichen Felder |
| Index für ISAM-Datei erzeugen | CI | Beschreibung der Index-Spalten (Key) |
| Index einer ISAM-Datei löschen | DI | Beschreibung der Index-Spalten (Key) |
| Datensätze physisch nach Schlüsselwerten sortieren (Cluster-Index) | CL | Beschreibung der Index-Spalten (Key) |

Die drei Protokollsatz-Typen, die Indizes betreffen (CI, DI und CL) enthalten genaue Information über die Index-Spalten (Keys). Format und Inhalt der Index-Spalten werden in folgenden Feldern ausgegeben:

```
key.k_nparts = n    key.k_flags = n  key.k_len = n
key.k_part:    kp_start     kp_leng    kp_type
```

key.k_nparts — Anzahl der Spalten des Index, wie sie in der Index-Beschreibung des Protokollsatzes enthalten sind.

key.k_flags — Anzahl der Index-Flags, die zur Index-Beschreibung des Protokollsatzes gehören.

key.k_len — Länge der Index-Beschreibung im Protokollsatz

key.k_part — beschreibt für jede Index-Spalte:

- kp_start — Anfangsbyte der Index-Spalte im gesamten (evtl.zusammengesetzten) Index
- kp_leng — Länge der Index-Spalte
- kp_type — Sortierreihenfolge der Index-Spalte

**Hinweise**

- Sie können die Schalter **-v**, **-l**, und **-h** zusammen angeben. Die Angabe der Position des Protokollsatzes (Schalter **-l**) wird vor allem während des Rücksetzens einer Transaktion mit ROLLBACK benutzt.
- Wenn Sie mit dem Schalter **-t** eine Anfangs- und Endezeit auswählen, ohne gleichzeitig mit dem Schalter **-d** ein Anfangs- und Endedatum auszuwählen, wird der aktuelle Tag verwendet.
- Mit dem Schalter **-f** geben Sie die gewünschte ISAM-Datei an, in der die Datenbank-Tabelle gespeichert ist. Den Namen der Datei können Sie mit folgender SQL-Anweisung abfragen:

```
SELECT dirpath
  FROM systables
  WHERE tabname = tabellenname
```

### Beispiele

Die folgenden Beispiele benützen die Protokolldatei *allcall.log.*

1. Alle Protokollsätze im Standardformat ausgeben:
   dblog allcall.log
2. Alle Protokollsätze mit Position und zusätzlicher Information ausgeben:
   dblog -l -v allcall.log
3. Alle Protokollsätze ausgeben, ohne zusätzliche Überschriften:
   dblog -h 0 allcall.log
4. Alle Protokollsätze ausgeben, mit einer Überschrift für jeden Protokollsatz:
   dblog -h 1 allcall.log
5. Alle Protokollsätze zwischen 9/11/89 und 9/20/89 ausgeben; Sätze vom Anfangs- und Endedatum sind in der Ausgabe enthalten.
   dblog -d 9/11/89 9/20/89 allcall.log
6. Alle Protokollsätze des heutigen Tages zwischen 10:01:01 und 15:59:59, einschließlich:
   dblog -t 10:01:01 15:59:59 allcall.log
7. Alle Protokollsätze ausgeben, die der Benutzer *informix* geschrieben hat:
   dblog -u informix allcall.log
8. Alle Protokollsätze ausgeben, die die C-ISAM-Datei *isfile* betreffen:
   dblog -f isfile allcall.log
9. Alle Protokollsätze ausgeben, der Position von 100 Bytes bis 550 Bytes in der Protokolldatei reicht.
   dblog -r 100 550 allcall.log

### Beispielausgaben

### Standardausgabe:

```
DBLOG: Transaction Log File Display C-ISAM Version 4.00.U
Copyright (C) 1981-1989 Informix Software, Inc.
Software Serial Number RDS#N000000

TY Trx ID User User Name    Date/Time      Lngth FD Recno Filename
-- ------ ---- ------------ -------------- ----- -- ----- ------------
BU    688  200 informix    9/ 8 17:16:55    37  0        isfile
BW    688  200 informix    9/ 8 17:16:55    20
FO    688  200 informix    9/ 8 17:16:55    29  0        isfile
IN    688  200 informix    9/ 8 17:16:55    36  0     1  isfile
FC    688  200 informix    9/ 8 17:16:55    29  0        isfile
CW    688  200 informix    9/ 8 17:16:55    20
BW    688  200 informix    9/ 8 17:16:56    20
FO    688  200 informix    9/ 8 17:16:56    29  0        isfile
IN    688  200 informix    9/ 8 17:16:56    36  0     2  isfile
FC    688  200 informix    9/ 8 17:16:56    29  0        isfile
RW    688  200 informix    9/ 8 17:16:56    20
BW    688  200 informix    9/ 8 17:16:56    20
FO    688  200 informix    9/ 8 17:16:56    29  0        isfile
CL    688  200 informix    9/ 8 17:16:56    28  0
FC    688  200 informix    9/ 8 17:16:57    33  0        issaa00688
CW    688  200 informix    9/ 8 17:16:57    20
BW    688  200 informix    9/ 8 17:16:58    20
FO    688  200 informix    9/ 8 17:16:58    29  0        isfile
IN    688  200 informix    9/ 8 17:16:58    36  0     2  isfile
IN    688  200 informix    9/ 8 17:16:58    36  0     3  isfile
```

```
TY Trx ID User User Name    Date/Time     Lngth FD Recno Filename
-- ------ ---- ------------ --------------- ----- -- ----- ------------
UP    688  200 informix     9/ 8 17:16:58    48  0     1 isfile
DE    688  200 informix     9/ 8 17:16:58    36  0     2 isfile
FC    688  200 informix     9/ 8 17:16:58    29  0       isfile
CW    688  200 informix     9/ 8 17:16:58    20
BW    688  200 informix     9/ 8 17:16:58    20
FO    688  200 informix     9/ 8 17:16:58    29  0       isfile
CI    688  200 informix     9/ 8 17:16:58    34  0
FC    688  200 informix     9/ 8 17:16:58    29  0       isfile
CW    688  200 informix     9/ 8 17:16:58    20
BW    688  200 informix     9/ 8 17:16:58    20
FO    688  200 informix     9/ 8 17:16:58    29  0       isfile
DI    688  200 informix     9/ 8 17:16:58    34  0
FC    688  200 informix     9/ 8 17:16:58    29  0       isfile
CW    688  200 informix     9/ 8 17:16:58    20
BW    688  200 informix     9/ 8 17:16:59    20
FO    688  200 informix     9/ 8 17:16:59    29  0       isfile
UN    688  200 informix     9/ 8 17:16:59    26  0     1
SU    688  200 informix     9/ 8 17:16:59    26  0   100
UN    688  200 informix     9/ 8 17:16:59    26  0   100
FC    688  200 informix     9/ 8 17:16:59    29  0       isfile

TY Trx ID User User Name    Date/Time     Lngth FD Recno Filename
-- ------ ---- ------------ --------------- ----- -- ----- ------------
CW    688  200 informix     9/ 8 17:16:59    20
RE    688  200 informix     9/ 8 17:16:59    39          isfile
ER    688  200 informix     9/ 8 17:17:00    28          isfile2

Program over.
```

## Ausgabe mit dblog -l

```
DBLOG: Transaction Log File Display C-ISAM Version 4.00.U
Copyright (C) 1981-1989 Informix Software, Inc.
Software Serial Number RDS#N000000
```

| TY | Trx ID | User | User Name | Date/Time | Lngth | FD | Recno | Filename | Location | Prev Loc | Pr Ln |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BU | 688 | 200 | informix | 9/ 8 17:16:55 | 37 | 0 | | isfile | 0 | 0 | 0 |
| BW | 688 | 200 | informix | 9/ 8 17:16:55 | 20 | | | | 37 | 0 | 0 |
| FO | 688 | 200 | informix | 9/ 8 17:16:55 | 29 | 0 | | isfile | 57 | 0 | 0 |
| IN | 688 | 200 | informix | 9/ 8 17:16:55 | 36 | 0 | 1 | isfile | 86 | 37 | 20 |
| FC | 688 | 200 | informix | 9/ 8 17:16:55 | 29 | 0 | | isfile | 122 | 0 | 0 |
| CW | 688 | 200 | informix | 9/ 8 17:16:55 | 20 | | | | 151 | 86 | 36 |
| BW | 688 | 200 | informix | 9/ 8 17:16:56 | 20 | | | | 171 | 151 | 20 |
| FO | 688 | 200 | informix | 9/ 8 17:16:56 | 29 | 0 | | isfile | 191 | 0 | 0 |
| IN | 688 | 200 | informix | 9/ 8 17:16:56 | 36 | 0 | 2 | isfile | 220 | 171 | 20 |
| FC | 688 | 200 | informix | 9/ 8 17:16:56 | 29 | 0 | | isfile | 256 | 0 | 0 |
| RW | 688 | 200 | informix | 9/ 8 17:16:56 | 20 | | | | 285 | 220 | 36 |
| BW | 688 | 200 | informix | 9/ 8 17:16:56 | 20 | | | | 305 | 285 | 20 |
| FO | 688 | 200 | informix | 9/ 8 17:16:56 | 29 | 0 | | isfile | 325 | 0 | 0 |
| CL | 688 | 200 | informix | 9/ 8 17:16:56 | 28 | 0 | | | 354 | 0 | 0 |
| FC | 688 | 200 | informix | 9/ 8 17:16:57 | 33 | 0 | | issaa006 | 382 | 0 | 0 |
| CW | 688 | 200 | informix | 9/ 8 17:16:57 | 20 | | | | 415 | 305 | 20 |
| BW | 688 | 200 | informix | 9/ 8 17:16:58 | 20 | | | | 435 | 415 | 20 |
| FO | 688 | 200 | informix | 9/ 8 17:16:58 | 29 | 0 | | isfile | 455 | 0 | 0 |
| IN | 688 | 200 | informix | 9/ 8 17:16:58 | 36 | 0 | 2 | isfile | 484 | 435 | 20 |
| IN | 688 | 200 | informix | 9/ 8 17:16:58 | 36 | 0 | 3 | isfile | 520 | 484 | 36 |

| TY | Trx ID | User | User Name | Date/Time | Lngth | FD | Recno | Filename | Location | Prev Loc | Pr Ln |
|---|---|---|---|---|---|---|---|---|---|---|---|
| UP | 688 | 200 | informix | 9/ 8 17:16:58 | 48 | 0 | 1 | isfile | 556 | 520 | 36 |
| DE | 688 | 200 | informix | 9/ 8 17:16:58 | 36 | 0 | 2 | isfile | 604 | 556 | 48 |
| FC | 688 | 200 | informix | 9/ 8 17:16:58 | 29 | 0 | | isfile | 640 | 0 | 0 |
| CW | 688 | 200 | informix | 9/ 8 17:16:58 | 20 | | | | 669 | 604 | 36 |
| BW | 688 | 200 | informix | 9/ 8 17:16:58 | 20 | | | | 689 | 669 | 20 |
| FO | 688 | 200 | informix | 9/ 8 17:16:58 | 29 | 0 | | isfile | 709 | 0 | 0 |
| CI | 688 | 200 | informix | 9/ 8 17:16:58 | 34 | 0 | | | 738 | 0 | 0 |
| FC | 688 | 200 | informix | 9/ 8 17:16:58 | 29 | 0 | | isfile | 772 | 0 | 0 |
| CW | 688 | 200 | informix | 9/ 8 17:16:58 | 20 | | | | 801 | 689 | 20 |
| BW | 688 | 200 | informix | 9/ 8 17:16:58 | 20 | | | | 821 | 801 | 20 |
| FO | 688 | 200 | informix | 9/ 8 17:16:58 | 29 | 0 | | isfile | 841 | 0 | 0 |
| DI | 688 | 200 | informix | 9/ 8 17:16:58 | 34 | 0 | | | 870 | 0 | 0 |
| FC | 688 | 200 | informix | 9/ 8 17:16:58 | 29 | 0 | | isfile | 904 | 0 | 0 |
| CW | 688 | 200 | informix | 9/ 8 17:16:58 | 20 | | | | 933 | 821 | 20 |
| BW | 688 | 200 | informix | 9/ 8 17:16:59 | 20 | | | | 953 | 933 | 20 |
| FO | 688 | 200 | informix | 9/ 8 17:16:59 | 29 | 0 | | isfile | 973 | 0 | 0 |
| UN | 688 | 200 | informix | 9/ 8 17:16:59 | 26 | 0 | 1 | | 1002 | 0 | 0 |
| SU | 688 | 200 | informix | 9/ 8 17:16:59 | 26 | 0 | 100 | | 1028 | 0 | 0 |
| UN | 688 | 200 | informix | 9/ 8 17:16:59 | 26 | 0 | 100 | | 1054 | 0 | 0 |
| FC | 688 | 200 | informix | 9/ 8 17:16:59 | 29 | 0 | | isfile | 1080 | 0 | 0 |

| TY | Trx ID | User | User Name | Date/Time | Lngth | FD | Recno | Filename | Location | Prev Loc | Pr Ln |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CW | 688 | 200 | informix | 9/ 8 17:16:59 | 20 | | | | 1109 | 953 | 20 |
| RE | 688 | 200 | informix | 9/ 8 17:16:59 | 39 | | | isfile | 1129 | 0 | 0 |
| ER | 688 | 200 | informix | 9/ 8 17:17:00 | 28 | | | isfile2 | 1168 | 0 | 0 |

```
Program over.
```

### Ausgabe bei dblog -v

```
DBLOG: Transaction Log File Display C-ISAM Version 4.00.U
Copyright (C) 1981-1989 Informix Software, Inc.
Software Serial Number RDS#N000000

TY Trx ID User User Name    Date/Time     Lngth FD Recno Filename
-- ------ ---- ------------ ------------- ----- -- ----- --------
BU    688  200 informix     9/ 8 17:16:55    37  0       isfile
    Row Length = 10    Build Mode = 0x806
BW    688  200 informix     9/ 8 17:16:55    20
FO    688  200 informix     9/ 8 17:16:55    29  0       isfile
IN    688  200 informix     9/ 8 17:16:55    36  0     1 isfile
    Image:
             0001 0000 0000 0000 0000                    ..........
FC    688  200 informix     9/ 8 17:16:55    29  0       isfile
CW    688  200 informix     9/ 8 17:16:55    20
BW    688  200 informix     9/ 8 17:16:56    20
FO    688  200 informix     9/ 8 17:16:56    29  0       isfile
IN    688  200 informix     9/ 8 17:16:56    36  0     2 isfile
    Image:
             0001 0000 0000 0000 0000                    ..........
FC    688  200 informix     9/ 8 17:16:56    29  0       isfile
RW    688  200 informix     9/ 8 17:16:56    20
BW    688  200 informix     9/ 8 17:16:56    20
FO    688  200 informix     9/ 8 17:16:56    29  0       isfile
CL    688  200 informix     9/ 8 17:16:56    28  0
FC    688  200 informix     9/ 8 17:16:57    33  0       issaa00688
CW    688  200 informix     9/ 8 17:16:57    20
BW    688  200 informix     9/ 8 17:16:58    20
FO    688  200 informix     9/ 8 17:16:58    29  0       isfile
IN    688  200 informix     9/ 8 17:16:58    36  0     2 isfile
    Image:
             0001 0000 0000 0000 0000                    ..........
IN    688  200 informix     9/ 8 17:16:58    36  0     3 isfile
    Image:
             0001 0000 0000 0000 0000                    ..........

TY Trx ID User User Name    Date/Time     Lngth FD Recno Filename
-- ------ ---- ------------ ------------- ----- -- ----- --------
UP    688  200 informix     9/ 8 17:16:58    48  0     1 isfile
    Pre-Image:
             0001 0000 0000 0000 0000                    ..........
    Post-Image:
             0002 0000 0000 0000 0000                    ..........
DE    688  200 informix     9/ 8 17:16:58    36  0     2 isfile
    Image:
             0001 0000 0000 0000 0000                    ..........
FC    688  200 informix     9/ 8 17:16:58    29  0       isfile
CW    688  200 informix     9/ 8 17:16:58    20
BW    688  200 informix     9/ 8 17:16:58    20
```

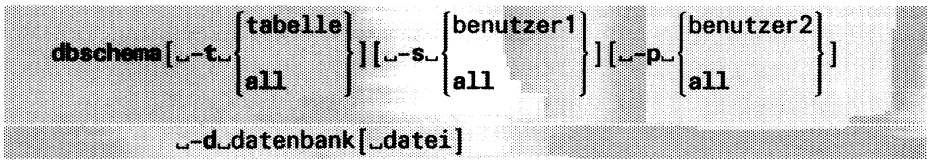
```
FO    688  200 informix    9/ 8 17:16:58    29  0       isfile
CI    688  200 informix    9/ 8 17:16:58    34  0
    key.k_nparts = 1    key.k_flags = 1    key.k_len = 2
    key.k_part:  kp_start  kp_leng  kp_type
                 --------  -------  -------
                        0        2        1
FC    688  200 informix    9/ 8 17:16:58    29  0       isfile
CW    688  200 informix    9/ 8 17:16:58    20
BW    688  200 informix    9/ 8 17:16:58    20
FO    688  200 informix    9/ 8 17:16:58    29  0       isfile
DI    688  200 informix    9/ 8 17:16:58    34  0
    key.k_nparts = 1    key.k_flags = 1    key.k_len = 2
    key.k_part:  kp_start  kp_leng  kp_type
                 --------  -------  -------
                        0        2        1
FC    688  200 informix    9/ 8 17:16:58    29  0       isfile
CW    688  200 informix    9/ 8 17:16:58    20
BW    688  200 informix    9/ 8 17:16:59    20
FO    688  200 informix    9/ 8 17:16:59    29  0       isfile
UN    688  200 informix    9/ 8 17:16:59    26  0     1
SU    688  200 informix    9/ 8 17:16:59    26  0   100
UN    688  200 informix    9/ 8 17:16:59    26  0   100
FC    688  200 informix    9/ 8 17:16:59    29  0       isfile

TY Trx ID User User Name    Date/Time     Lngth FD Recno Filename
-- ------ ---- ------------ ------------- ----- -- ----- ------------
CW    688  200 informix    9/ 8 17:16:59    20
RE    688  200 informix    9/ 8 17:16:59    39          isfile
    New Filename = isfile2
ER    688  200 informix    9/ 8 17:17:00    28          isfile2
BU    699 4116 jac         9/ 8 17:18:27    37  0       isfile
      Row Length = 10    Build Mode = 0x806

Program over.
```

### A.2.6 dbschema - Anweisungen für Datenbank-Aufbau erzeugen

*dbschema* erzeugt eine Anweisungsdatei, die CREATE TABLE-, CREATE INDEX- und CREATE VIEW-Anweisungen enthält, die für den Aufbau einer Datenbank oder einer bestimmten Tabelle nötig sind. Außerdem kann *dbschema* alle CREATE SYNONYM- und GRANT-Anweisungen erzeugen, die für eine Datenbank, eine bestimmte Tabelle oder einen View definiert sind.

Standardmäßig erzeugt *dbschema* alle CREATE TABLE-, CREATE VIEW-, CREATE INDEX-, CREATE SYNONYM- und GRANT-Anweisungen für die gesamte Datenbank. Durch Auswahl der entsprechenden Operanden können Sie die Ausgabe beschränken – auf eine bestimmte Tabelle oder View bzw. – auf Synonyme und Zugriffsrechte für einen bestimmten Benutzer.

*dbschema* setzt in die erzeugten Anweisungen jeweils den aktuellen Benutzernamen des Eigentümers ein. Wenn Sie für den erneuten Datenbank-Aufbau andere Eigentümer einsetzen wollen, so müssen Sie die Anweisungsdatei vor Ablauf entsprechend ändern.




**-t** tabelle

Name der Tabelle oder des Views, für die CREATE TABLE (bzw. CREATE VIEW) und CREATE INDEX-Anweisungen erzeugt werden sollen.

**all**

bezeichnet alle Tabellen und Views in der Datenbank.

**-s** benutzer1

Name des Benutzers, für den die CREATE SYNONYM-Anweisungen erzeugt werden sollen. Ist der Schalter **-t** angegeben, so werden CREATE SYNONYM-Anweisungen nur für die dort angegebene Tabelle bzw. View erzeugt. Ohne Schalter **-t** werden alle CREATE SYNONYM-Anweisungen erzeugt, die der angegebene Benutzer in der Datenbank gegeben hat.

**all**

bezeichnet alle Benutzer.

**-p** benutzer2

Name des Benutzer, für den die GRANT-Anweisungen erzeugt werden sollen. Ist der Schalter **-t** angegeben, so werden GRANT-Anweisungen nur für die dort angegebene Tabelle bzw. View erzeugt. Ohne Schalter **-t** werden alle GRANT-Anweisungen erzeugt, die der angegebene Benutzer in der Datenbank gegeben hat. Die GRANT-Anweisung enthält auch den Namen des Benutzers, der die GRANT-Anweisung gegeben hat (GRANT ... AS benutzer).

Für den späteren Aufruf der Anweisungsdatei müssen folgende Voraussetzungen erfüllt sein, damit die enthaltenen GRANT-Anweisungen ablaufen können:

Der Benutzer (AS benutzer), anstelle dessen das Zugriffsrecht vergeben wird, muß das Datenbankzugriffsrecht CONNECT haben und, falls er nicht Eigentümer der Tabelle ist, das entsprechende Tabellenzugriffsrecht inklusive der Berechtigung WITH GRANT OPTION.

**all**

bezeichnet alle Benutzer.

**-d** datenbank

Name der Datenbank. Für INFORMIX-SE gilt: Ist die Datenbank nicht im aktuellen Dateiverzeichnis vorhanden, durchsucht *dbschema* die Dateiverzeichnisse, die in der Umgebungsvariablen DBPATH angegeben sind. *dbschema* wird abgewiesen, wenn die Datenbank nicht gefunden wird.

datei

Name der Datei, in die *dbschema* die erzeugten Anweisungen schreiben soll. Fehlt die Angabe, so werden die Anweisungen auf dem Bildschirm ausgegeben.

### Hinweis

Spalten vom Datentyp SERIAL erhalten in der CREATE TABLE-Anweisung den Anfangswert 1, unabhängig von ihrem ursprünglichen Anfangswert.

### Beispiel

Für die Datenbank *versand* werden die Anweisungen erzeugt, die zum Aufbau der Tabelle *auftrag* und deren Indizes nötig sind:

```
dbschema -t auftrag -d versand
```

```
DBSCHEMA Schema Utility       INFORMIX-SQL Version 4.00.U
Copyright (C) Informix Software, Inc., 1984-1989
Software Serial Number RDS#N000000
{ TABLE "lomata".auftrag row size = 80 number of columns = 10 index size = 24 }
create table "lomata".auftrag
  (
    auftrags_nr serial not null,
    auftragsdatum date,
    kunden_nr integer,
    lieferhinweis char(40),
    offen char(1),
    fremd_nr char(10),
    lieferdatum date,
    liefergewicht decimal(8,2),
    zustellgebuehr money(6,2),
    zahldatum date
  );
revoke all on "lomata".auftrag from "public";

create unique index "lomata".a_nr_ix on "lomata".auftrag (auftrags_nr);
create index "lomata".a_knr_ix on "lomata".auftrag (kunden_nr);
```

# A.3 ASCII-Tabelle

| dezi-mal | oktal | hexa-dez. | | Bedeutung | Control |
|---|---|---|---|---|---|
| 0 | 00 | 00 | NUL | Null, keine Operation | @ |
| 1 | 01 | 01 | SOH | Start of Heading<br>Vorspannanfang | A |
| 2 | 02 | 02 | STX | Start of Text<br>Textanfang | B |
| 3 | 03 | 03 | ETX | End of Text<br>Textende | C |
| 4 | 04 | 04 | EOT | End of Transmission Taste END<br>Übertragungsende | D |
| 5 | 05 | 05 | ENQ | Enquiry<br>Stationsanruf | E |
| 6 | 06 | 06 | ACK | Acknowledge<br>Bestätigung | F |
| 7 | 07 | 07 | BEL | Bell<br>Klingel | G |
| 8 | 10 | 08 | BS | Backspace<br>Korrekturtaste | H |
| 9 | 11 | 09 | HT | Horizontal Tabulation<br>Tabulatorzeichen | I |
| 10 | 12 | 0A | LF | Line Feed<br>Zeilenvorschub, neue Zeile | J |
| 11 | 13 | 0B | VT | Vertical Tabulation | K |
| 12 | 14 | 0C | FF | Form Feed<br>Formularvorschub | L |
| 13 | 15 | 0D | CR | Carriage Return<br>Wagenrücklauf | M |
| 14 | 16 | 0E | SO | Shift Out<br>Umschalten Zeichensatz | N |
| 15 | 17 | 0F | SI | Shift In<br>Zurückschalten Zeichensatz | O |
| 16 | 20 | 10 | DLE | Data Link Escape<br>Austritt aus der Datenverbindung | P |
| 17 | 21 | 11 | DC1 | Device Control 1<br>Gerätesteuerung 1, Ausgabe fortsetzen | Q |
| 18 | 22 | 12 | DC2 | Device Control 2 | R |
| 19 | 23 | 13 | DC3 | Device Control 3 Ausgabe anhalten | S |
| 20 | 24 | 14 | DC4 | Device Control 4 | T |
| 21 | 25 | 15 | NAK | Negative Acknowledge<br>Fehlermeldung | U |
| 22 | 26 | 16 | SYN | Synchronous Idle<br>Synchronisierung | V |
| 23 | 27 | 17 | ETB | End of Transm. Block<br>Datenblockende | W |
| 24 | 30 | 18 | CAN | Cancel<br>ungültig, Zeilenlöscher | X |
| 25 | 31 | 19 | EM | End of Medium<br>Datenträgerende, quit (Signal3) | Y |
| 26 | 32 | 1A | SUB | Substitute Character<br>Zeichen ersetzen | Z |

| dezi-mal | oktal | hexa-dez. | | Bedeutung | | Control |
|---|---|---|---|---|---|---|
| 27 | 33 | 1B | ESC | Escape | Rücksprung | |
| 28 | 34 | 1C | FS | File Separator | Dateitrennung | \ |
| 29 | 35 | 1D | GS | Group Separator | Gruppentrennung | ] |
| 30 | 36 | 1E | RS | Record Separator | Satztrennung | ^ |
| 31 | 37 | 1F | US | Unit Separator | Einheitentrennung | _ oder DEL |
| 32 | 40 | 20 | SP | SPACE | Leerzeichen | |
| 33 | 41 | 21 | ! | | | |
| 34 | 42 | 22 | " | | | |
| 35 | 43 | 23 | # | Nummernzeichen | | |
| 36 | 44 | 24 | $ | oder nationales Währungssymbol | | |
| 37 | 45 | 25 | % | | | |
| 38 | 46 | 26 | & | | | |
| 39 | 47 | 27 | ' | | | |
| 40 | 50 | 28 | ( | | | |
| 41 | 51 | 29 | ) | | | |
| 42 | 52 | 2A | * | (Stern gilt oft als Multiplikations-zeichen) | | |
| 43 | 53 | 2B | + | | | |
| 44 | 54 | 2C | , | | | |
| 45 | 55 | 2D | - | | | |
| 46 | 56 | 2E | . | | | |
| 47 | 57 | 2F | / | (dient als Divisionszeichen) | | |
| 48 | 60 | 30 | 0 | | | |
| 49 | 61 | 31 | 1 | | | |
| 50 | 62 | 32 | 2 | | | |
| 51 | 63 | 33 | 3 | | | |
| 52 | 64 | 34 | 4 | | | |
| 53 | 65 | 35 | 5 | | | |
| 54 | 66 | 36 | 6 | | | |
| 55 | 67 | 37 | 7 | | | |
| 56 | 70 | 38 | 8 | | | |
| 57 | 71 | 39 | 9 | | | |
| 58 | 72 | 3A | : | | | |
| 59 | 73 | 3B | ; | | | |
| 60 | 74 | 3C | < | | | |
| 61 | 75 | 3D | = | | | |
| 62 | 76 | 3E | > | | | |
| 63 | 77 | 3F | ? | | | |
| 64 | 100 | 40 | @ | (kaufmännisches "at" oder §) | | |
| 65 | 101 | 41 | A | | | |
| 66 | 102 | 42 | B | | | |
| 67 | 103 | 43 | C | | | |
| 68 | 104 | 44 | D | | | |
| 69 | 105 | 45 | E | | | |
| 70 | 106 | 46 | F | | | |
| 71 | 107 | 47 | G | | | |
| 72 | 110 | 48 | H | | | |
| 73 | 111 | 49 | I | | | |
| 74 | 112 | 4A | J | | | |
| 75 | 113 | 4B | K | | | |
| 76 | 114 | 4C | L | | | |
| 77 | 115 | 4D | M | | | |

| dezi-mal | oktal | hexa-dez. | | Bedeutung | Control |
|---|---|---|---|---|---|
| 78 | 116 | 4E | N | | |
| 79 | 117 | 4F | O | | |
| 80 | 120 | 50 | P | | |
| 81 | 121 | 51 | Q | | |
| 82 | 122 | 52 | R | | |
| 83 | 123 | 53 | S | | |
| 84 | 124 | 54 | T | | |
| 85 | 125 | 55 | U | | |
| 86 | 126 | 56 | V | | |
| 87 | 127 | 57 | W | | |
| 88 | 130 | 58 | X | | |
| 89 | 131 | 59 | Y | | |
| 90 | 132 | 5A | Z | | |
| 91 | 133 | 5B | [ | oder Ä | |
| 92 | 134 | 5C | \ | Gegenschrägstrich oder Ö | |
| 93 | 135 | 5D | ] | oder Ü | |
| 94 | 136 | 5E | ^ | oder ↑ | |
| 95 | 137 | 5F | _ | Unterstrich oder r | |
| 96 | 140 | 60 | ` | | |
| 97 | 141 | 61 | a | | |
| 98 | 142 | 62 | b | | |
| 99 | 143 | 63 | c | | |
| 100 | 144 | 64 | d | | |
| 101 | 145 | 65 | e | | |
| 102 | 146 | 66 | f | | |
| 103 | 147 | 67 | g | | |
| 104 | 150 | 68 | h | | |
| 105 | 151 | 69 | i | | |
| 106 | 152 | 6A | j | | |
| 107 | 153 | 6B | k | | |
| 108 | 154 | 6C | l | | |
| 109 | 155 | 6D | m | | |
| 110 | 156 | 6E | n | | |
| 111 | 157 | 6F | o | | |
| 112 | 160 | 70 | p | | |
| 113 | 161 | 71 | q | | |
| 114 | 162 | 72 | r | | |
| 115 | 163 | 73 | s | | |
| 116 | 164 | 74 | t | | |
| 117 | 165 | 75 | u | | |
| 118 | 166 | 76 | v | | |
| 119 | 167 | 77 | w | | |
| 120 | 170 | 78 | x | | |
| 121 | 171 | 79 | y | | |
| 122 | 172 | 7A | z | | |
| 123 | 173 | 7B | { | oder ä | |
| 124 | 174 | 7C | \| | oder ö | |
| 125 | 175 | 7D | } | oder ü | |
| 126 | 176 | 7E | ~ | oder ß | |
| 127 | 177 | 7F | DEL | Delete<br>Löschzeichen, Interrupt (Signal2) | |

**Hinweis**: Verwendung von Umlauten bei INFORMIX

Benutzer, die mit der Taste CH CODE ihre Bildschirmdarstellung von internationalem auf deutschen Zeichensatz umschalten können, können bei INFORMIX auch mit Umlauten arbeiten. Die Umlaute belegen jedoch denselben ASCII-Code wie Zeichen, die INFORMIX als Steuerzeichen interpretiert, nämlich:

| | | |
|---|---|---|
| Ä | entspricht dem Zeichen | [ |
| ä | entspricht dem Zeichen | { |
| Ü | entspricht dem Zeichen | ] |
| ü | entspricht dem Zeichen | } |
| Ö | entspricht dem Zeichen | \ |
| ö | entspricht dem Zeichen | \| |
| ß | entspricht dem Zeichen | ~ |

Sie müssen daher folgendes beachten, wenn Sie Umlaute verwenden:

- Verwenden Sie keine Umlaute in Kommentaren, die durch geschweifte Klammern { } begrenzt werden.
- Definieren Sie die Standard-Feldbegrenzer [] von Formaten um, z.B: in <>.
- Definieren Sie das Feldtrennzeichen '|' in LOAD/UNLOAD-Dateien um (Umgebungsvariable DBDELIMITER).
- Definieren Sie für die Ausgabe den deutschen Zeichensatz: lpr -dt (Umgebungsvariable DBPRINT).
- Definieren Sie bei Formaten keine Felder mit der Feldeigenschaft UPSHIFT oder DOWNSHIFT (da die Groß/Klein-Umwandlung gemäß ASCII-Tabelle zu keinem sinnvollen Ergebnis führt).
- Beachten Sie, daß beim Sortieren in Reihenfolge des ASCII-Codes sortiert wird.

Weitere Informationen zur Umlautverwendung enthält Kapitel 9.5 des Handbuchs *INFORMIX Kennenlernen* [2].

## Literatur

Die mit * gekennzeichneten Titel sind nicht von Siemens Nixdorf Informationssysteme AG herausgegeben.

[1] Betriebssystem SINIX
**INFORMIX**
SQL-Sprachbeschreibung

[2] Betriebssystem SINIX
**INFORMIX-SQL**
Kennenlernen

[3] Betriebssystem SINIX
**Fehlermeldungen für**
**INFORMIX-Produkte**

[4] Betriebssystem SINIX
**INFORMIX-ONLINE**
Datenbank-Server
Administrator-Handbuch

[5] Betriebssystem SINIX
**INFORMIX-NET**
Netzkomponente für INFORMIX-SE

[6] Betriebssystem SINIX
**INFORMIX-STAR**
Netzkomponente für INFORMIX-ONLINE

[7] Betriebssystem SINIX
**INFORMIX-ESQL/C**
C-Schnittstelle für das
Datenbanksystem INFORMIX

[8] Betriebssystem SINIX
**INFORMIX-ESQL/COBOL**
COBOL-Schnittstelle für das
Datenbanksystem INFORMIX

[9] Betriebssystem SINIX
**INFORMIX-4GL**
Nachschlagen

[10] Betriebssystem SINIX
**INFORMIX-4GL**
Kennenlernen

[11] Betriebssystem SINIX
**C-ISAM**
Indexsequentielle Zugriffsmethode

[12] Betriebssystem SINIX
**Systemverwaltung**

[13] Betriebssystem SINIX
**Kommandos**

[14] Betriebssystem SINIX
**CES**
C-Entwicklungsssytem

[15] Betriebsssytem SINIX
**Schnittstellen**

* C.J. Date
An Introduction to Database Systems
Addison-Wesley
1986

* C.J. Date
A Guide to The SQL Standard
Addison-Wesley
1990

### Bestellen von Handbüchern

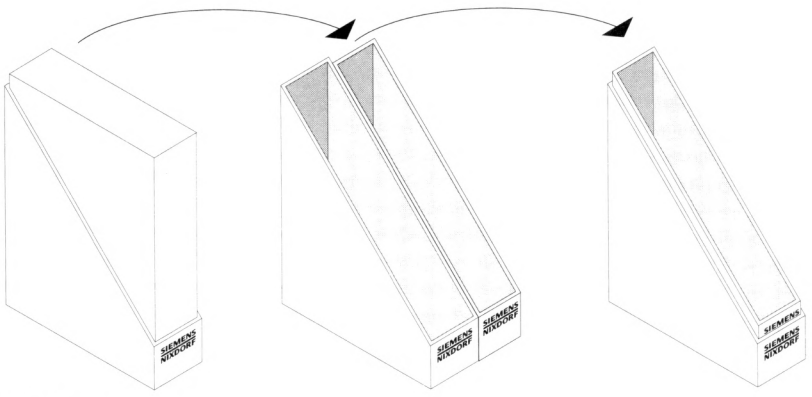
Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriftenverzeichnis Datentechnik.* Dort ist auch der Bestellvorgang erklärt. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen Datentechnik.*

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an eine Geschäftsstelle unseres Hauses.

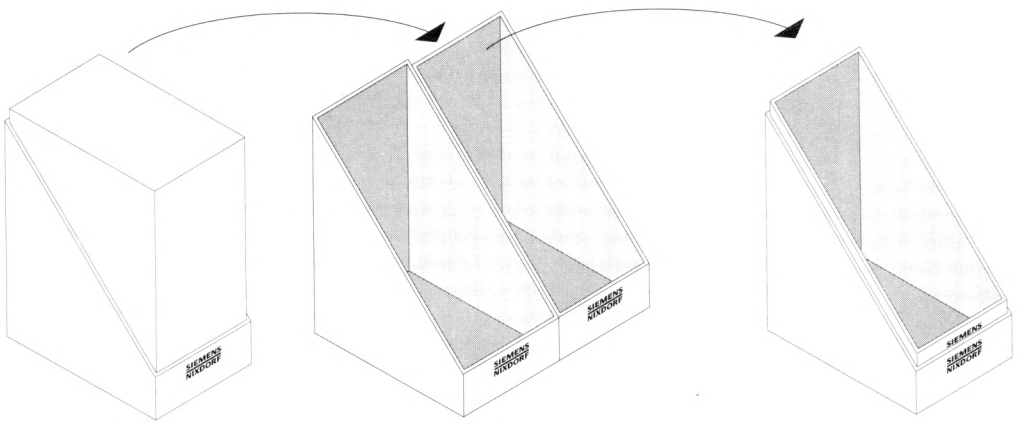
# Stichwortverzeichnis

# Sammelboxen

Für Handbücher des vorliegenden Formates bieten wir zweiteilige Sammelboxen in zweierlei Größen an. Der Bestellvorgang entspricht dem für Handbücher.




Breite: ca. 5 cm
Bestellnummer: U3775-J-Z18-1




Breite: ca. 10 cm
Bestellnummer: U3776-J-Z18-1


970071 P/MCD

Herausgegeben von/Published by
Siemens Nixdorf Informationssysteme AG
Postfach 21 60, W-4790 Paderborn
Postfach 83 09 51, W-8000 München 83

Bestell-Nr./Order No. ***U2605-J-Z95-3***
Printed in the Federal Republic of Germany
6190 AG 11903. (7740)

SIEMENS
NIXDORF

SINIX

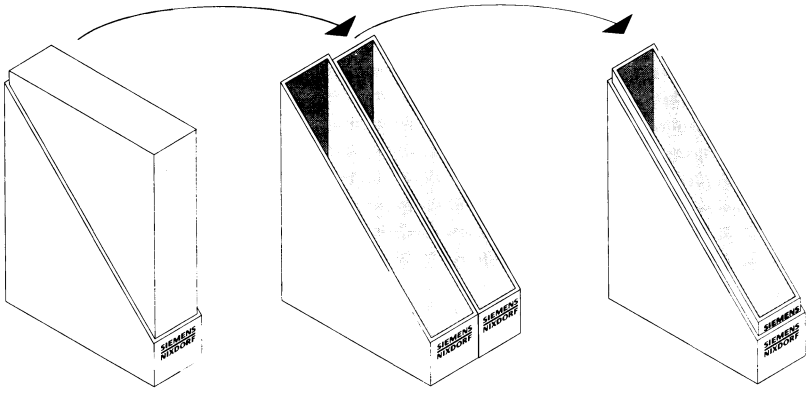
# INFORMIX-SQL V4.0

## Nachschlagen

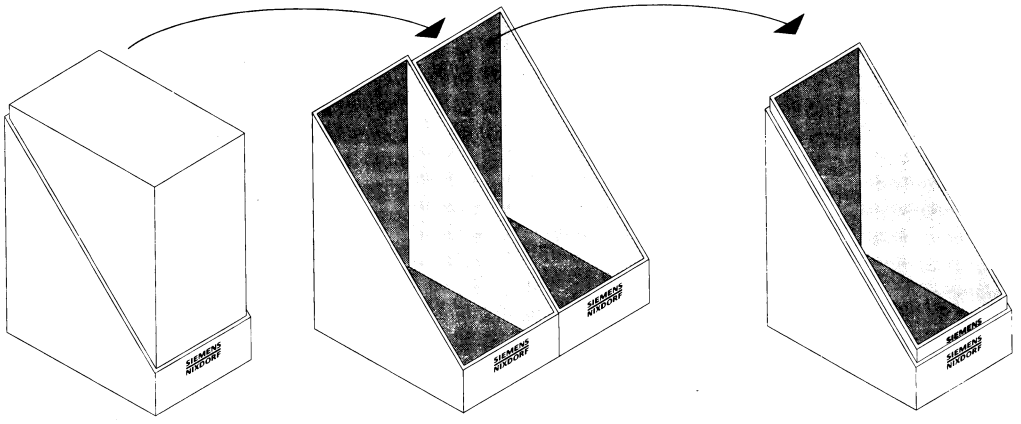
Benutzerhandbuch

# Sammelboxen

Für Handbücher des vorliegenden Formates bieten wir zweiteilige Sammelboxen in zweierlei Größen an. Der Bestellvorgang entspricht dem für Handbücher.




Breite: ca. 5 cm
Bestellnummer: U3775-J-Z18-1




Breite: ca. 10 cm
Bestellnummer: U3776-J-Z18-1


970071 P/MCD

Herausgegeben von/Published by
Siemens Nixdorf Informationssysteme AG
Postfach 21 60, W-4790 Paderborn
Postfach 83 09 51, W-8000 München 83

Bestell-Nr./Order No. ***U2605-J-Z95-3***
Printed in the Federal Republic of Germany
6190 AG 11903. (7740)